

HP 75000 SERIES C

Command Module HP E1405B

User's Manual

Copyright © Hewlett-Packard Company, 1991



Manual Part Number: E1405-90004
Microfiche Part Number: E1405-99004

Printed: October 1991
Printed in U.S.A.

Edition 4
E 1091

1

2

3



HP 75000 SERIES C

**Command Module
HP E1405B**

User's Manual



Copyright © Hewlett-Packard Company, 1991

Manual Part Number: E1405-90004
Microfiche Part Number: E1405-99004

Printed: October 1991
Printed in U.S.A.

Edition 4
E 1091

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (formerly National Bureau of Standards), to the extent allowed by that organization's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard product is warranted against defects in materials and workmanship for a period of three years from date of shipment. Duration and conditions of warranty for this product may be superceded when the product is integrated into (becomes a part of) other HP products. During the warranty period, Hewlett-Packard Company will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Hewlett-Packard (HP). Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with a product will execute its programming instructions when properly installed on that product. HP does not warrant that the operation of the product, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied products or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

The design and implementation of any circuit on this product is the sole responsibility of the Buyer. HP does not warrant the Buyer's circuitry or malfunctions of HP products that result from the Buyer's circuitry. In addition, HP does not warrant any damage that occurs as a result of the Buyer's circuit or any defects that result from Buyer-supplied products.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

NOTICE

The information contained in this document is subject to change without notice. HEWLETT-PACKARD (HP) MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HP shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Hewlett-Packard Company. HP assumes no responsibility for the use or reliability of its software on equipment that is not furnished by HP.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (b)(3)(ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013. Hewlett-Packard Company; 3000 Hanover Street; Palo Alto, California 94304

Declaration of Conformity

According to ISO/IEC Guide 22 and EN 45014

The Hewlett-Packard Company declares that the HP E1445A conforms to the following Product Specifications.

Safety: IEC 1010
CSA 231
UL 1244

EMC: CISPR 11, EN 55011, Class A
IEC 801-2, EN 50082-1, 4kVCD, 8kVAD
IEC 801-3, EN 50082-1, 3 V/M
IEC 801-4, EN 50082-1, 1kV


Q.A. Manager
November 1991

Hewlett-Packard Company
P.O. Box 301
815 14th Street S.W.
Loveland, Colorado 80539 U.S.A

Printing History

The Printing History shown below lists all Editions and Updates of this manual and the printing date(s). The first printing of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct the current Edition of the manual. Updates are numbered sequentially starting with Update 1. When a new Edition is created, it contains all the Update information for the previous Edition. Each new Edition or Update also includes a revised copy of this printing history page. Many product updates or revisions do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

Edition 4 (Part Number E1405-90004) October 1991

Trademark Information

Microsoft® and MS-DOS® are U.S. registered trademarks of Microsoft Corporation.
IBM® and PC-DOS® are U.S. registered trademarks of International Business Machines Corporation.
DEC®, VT100®, and VT220® are registered trademarks of Digital Equipment Corporation.
WYSE® is a registered trademark of Wyse Technology
WY-30™ is a trademark of Wyse Technology
Macintosh® is a registered trademark of Apple Computer Inc.

Safety Symbols



Instruction manual symbol affixed to product. Indicates that the user must refer to the manual for specific Warning or Caution information to avoid personal injury or damage to the product.



Indicates the field wiring terminal that must be connected to earth ground before operating the equipment—protects against electrical shock in case of fault.



OR Frame or chassis ground terminal—typically connects to the equipment's metal frame.



Alternating current (AC).



Direct current (DC).



Indicates hazardous voltages.

WARNING

Calls attention to a procedure, practice, or condition that could cause bodily injury or death.

CAUTION

Calls attention to a procedure, practice, or condition that could possibly cause damage to equipment or permanent loss of data.

WARNINGS

The following general safety precautions must be observed during all phases of operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the product. Hewlett-Packard Company assumes no liability for the customer's failure to comply with these requirements.

Ground the equipment: For Safety Class 1 equipment (equipment having a protective earth terminal), an uninterruptible safety earth ground must be provided from the mains power source to the product input wiring terminals or supplied power cable.

DO NOT operate the product in an explosive atmosphere or in the presence of flammable gases or fumes.

For continued protection against fire, replace the line fuse(s) only with fuse(s) of the same voltage and current rating and type. **DO NOT** use repaired fuses or short-circuited fuses.

Keep away from live circuits: Operating personnel must not remove equipment covers or shields. Procedures involving the removal of covers or shields are for use by service-trained personnel only. Under certain conditions, dangerous voltages may exist even with the equipment switched off. To avoid dangerous electrical shock, **DO NOT** perform procedures involving cover or shield removal unless you are qualified to do so.

DO NOT operate damaged equipment: Whenever it is possible that the safety protection features built into this product have been impaired, either through physical damage, excessive moisture, or any other reason, **REMOVE POWER** and do not use the product until safe operation can be verified by service-trained personnel. If necessary, return the product to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

DO NOT service or adjust alone: Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT substitute parts or modify equipment: Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

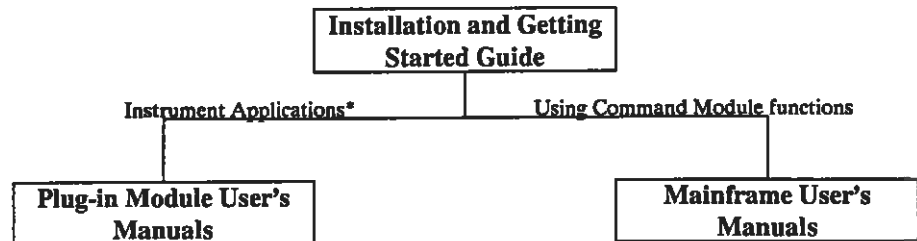
HP 75000 Series C Documentation

Suggested Sequence for Using the Manuals

C-Size VXIbus Systems Installation and Getting Started Guide. Contains step-by-step instructions for all aspects of plug-in module and mainframe installation.

HP E1405A Command Module User's Manual. Contains information on downloading user tables to modify (if necessary) configurations set up using the Installation and Getting Started Guide, information on using an RS-232 terminal as a "front panel" to your C-size system, and information on how interrupts are used. A command reference for the E1405 Command Module command set is included.

Plug-In Module User's Manuals. Contain programming and configuration information for the plug-in modules. These manuals contain examples for the most commonly-used functions and give a complete SCPI command reference for the module.



* For Scanning Voltmeter Applications, refer to the HP E1326A/E1411A 5 1/2 Digit Multimeter User's Manual.

Suggested Sequence for Using the Manuals

Related Documents

HP E1400B Mainframe User's Manual. Contains installation information to prepare the mainframe for use and shows how to install plug-in manuals. This manual also contains a detailed hardware description of the mainframe.

HP Instrument BASIC User's Handbook. Includes three books: *HP Instrument BASIC Programming Techniques*, *HP Instrument BASIC Interfacing Techniques*, and *HP Instrument BASIC Language Reference*.

Using HP Instrument BASIC with the E1405. Contains information on the version of HP Instrument Basic which can be installed in ROM in your E1405B Command Module.

Beginner's Guide to SCPI. Explains the fundamentals of programming instruments using the Standard Commands for Programmable Instruments (SCPI) language. We recommend this guide to anyone who is programming with SCPI for the first time.

Tutorial Description of the Hewlett-Packard Interface Bus. Describes the technical fundamentals of the Hewlett-Packard Interface Bus (HP-IB). This document also includes general information on IEEE 488.2 Common Commands. We recommend this document to anyone who is programming with IEEE 488.2 for the first time.

IEEE Standard 488.2-1987, IEEE Standard Codes, Formats, Protocols, and Common Commands. Describes the underlying message formats and data types used in TMSL and defines Common Commands. You will find this document useful if you need to know the precise definition of certain message formats, data types, or Common Commands. Available from: The Institute of Electrical and Electronic Engineers, Inc.; 345 East 47th Street; New York, NY 10017; U.S.A.

VXIbus System Specifications. Available from Hewlett-Packard.

The VMEbus Specification. Available from: VMEbus International Trade Association; 10229 N. Scottsdale Road, Suite E; Scottsdale, AZ 85253; U.S.A.

About this Manual

Manual Content

- This manual contains information on the applications of the HP E1405B Command Module. The manual is part of a manual set that includes the C-Size VXIbus Systems "Installation and Getting Started Guide" and various plug-in module user's manuals.
- Chapter 1: Product Overview** This chapter contains a functional, electrical, and physical description of the HP E1405B Command Module.
- Chapter 2: Modifying Your Configuration** This chapter explains how the Command Module's resource manager function configures your VXIbus system. It also contains information on using user-tables to override the (default) configuration performed by the resource manager.
- Chapter 3: Using the Display Terminal Interface** This chapter shows you how to use an RS-232 terminal to operate instruments in the Series C mainframe. The terminal is connected to the Command Module via the Module's RS-232 port.
- Chapter 4: Status and Interrupts** This chapter describes the status system structure used by the Command Module and how interrupts are enabled and serviced.
- Chapter 5: Downloading Device Drivers** This chapter contains information on downloading device drivers into non-volatile memory using both HP-IB and RS-232 connections.
- Chapter 6: E1405 Command Reference** The command reference contains a detailed description of each Command Module command. It includes information on the choice of settings and examples showing the context in which the command is used.
- Appendix A: Specifications** This section contains a list of the HP E1405 Command Module's operating specifications.
- Appendix B: Error Messages** This section lists the error messages associated with the Command Module and their possible causes.
- Appendix C: Command Module A16 Address Space** This appendix contains an address map of the A16 address space inside the Command Module. It includes information on how to determine the base address of a device whose registers are mapped into A16 space.
- Appendix D: Sending Binary Data Over RS-232** This Appendix contains information on transferring binary files over an RS-232 interface. It includes information on how these files are coded for transmission.

Table of Contents

1. Product Overview

About this Chapter	1-1
Command Module Functional Description	1-1
Command Module Electrical Description	1-2
Command Module Physical Description	1-2
Faceplate Annunciators	1-2
Faceplate CLK10 and Trigger Connectors	1-2
The HP-IB and RS-232 Ports	1-3
The Reset Button	1-3
Installing/Removing the Command Module	1-3
Command Module Memory	1-3
Battery Backed Functions	1-3

2. Modifying Your Configuration

About this Chapter	2-1
System Configuration Sequence	2-1
Identifying Statically and Dynamically Configured Modules	2-2
Statically Configured (SC) Modules	2-2
Dynamically Configured (DC) Modules	2-2
User-Defined Dynamic Configuration	2-2
Setting Up VXI-MXI Configuration	2-7
Logical Address Configuration	2-7
A16/A24/A32 Address Window Configuration	2-8
Interrupt Register Configuration	2-9
TTL Trigger Register Configuration	2-9
ECL Trigger Register Configuration	2-10
Utility Register Configuration	2-10
User Defined Logical Address and Memory Windows	2-10
Setting Commander/Servant Hierarchies	2-17
User-Defined Commander/Servant Hierarchies	2-18
Example: Assigning a Secondary HP-IB Address	2-20
A24/A32 Address Mapping	2-22
Reserving A24/A32 Address Space	2-26
Example: Reserving A24 Addresses for a VME Device	2-28
Interrupt Line Allocation	2-30
User-Defined Interrupt Line Allocation	2-32
Example: Assigning an Interrupt Line	2-35
Starting System Operation	2-37

3. Using the Display Terminal Interface

About this Chapter	3-1
Terminal Interface Features	3-1
Using Menus	3-2
How Instruments Appear in the Menu	3-3

A 60-Second Menu Tutorial	3-3
Using the System Instrument Menu	3-5
Using the Other Instrument Menus	3-12
Monitor Mode	3-15
Executing Commands	3-17
Editing	3-17
General Key Descriptions	3-18
Menu and Menu Control Keys	3-18
Editing Keys	3-18
Instrument Control Keys	3-19
Other Keys	3-19
Using Supported Terminals	3-20
The Supported Terminals	3-20
Using the HP 700/22	3-21
Using the WYSE® WY-30ce	3-23
Using Other Terminals	3-24
What "Not Supported" Means	3-24
Testing Terminals for Compatibility	3-24
Using a Terminal Without Menus	3-25
In Case of Difficulty	3-27
Instrument Menus	3-29

4. Status and Interrupts

About this Chapter	4-1
Status System Structure	4-2
The Status Byte Register	4-3
Reading the Status Byte Register	4-4
Service Request Enable Register	4-5
The Service Request Bit	4-5
Clearing the Service Request Enable Register	4-5
Standard Event Status Register	4-6
Unmasking Standard Event Status Bits	4-6
Reading the Standard Event Status Enable Register Mask	4-7
Reading the Standard Event Status Register	4-7
Operation Status Group	4-7
Reading the Condition Register	4-8
Unmasking the Operation Event Register Bits	4-8
Clearing the Operation Event Register Bits	4-9
Using the Operation Status Group Registers	4-9
Clearing Status	4-10
Interrupting an External Computer	4-10
Synchronizing an External Computer and Instruments	4-12

5. Downloading Device Drivers

About this Chapter	5-1
What You Will Need	5-1
Memory Configuration	5-3
Download Program Configuration	5-4
Editing the Configuration File	5-4

Downloading Drivers in MS-DOS Systems	5-6
Downloading Drivers in HP-IB Systems with IBASIC	5-7
Downloading Drivers in HP-IB Systems with HP BASIC	5-8
Downloading Multiple Drivers	5-9
Checking Driver Status	5-9
Manually Downloading a Driver	5-10
Preparing Memory for Manual Downloading	5-10
Manually Downloading Over HP-IB	5-11
Manually Downloading Over RS-232	5-11

6. E1405 Command Reference

About This Chapter	6-1
Command Types	6-1
Common Command Format	6-1
SCPI Command Format	6-1
Linking Commands	6-3
SCPI Command Reference	6-4
DIAGnostic	6-4
OUTPut	6-27
STATus	6-35
SYSTem	6-38
VXI 6-54	
Common Command Reference	6-83
*CLS	6-84
*DMC < name_string >, < command_block >	6-84
*EMC < enable >	6-84
*EMC?	6-84
*ESE < mask >	6-84
*ESE?	6-85
*ESR?	6-85
*GMC? < name_string >	6-85
*IDN?	6-86
*LMC?	6-86
*LRN?	6-86
*OPC	6-87
*OPC?	6-87
*PMC	6-87
*PSC < flag >	6-87
*PSC?	6-87
*RMC < name_string >	6-88
*RST	6-88
*SRE < mask >	6-88
*SRE?	6-88
*STB?	6-89
*TST?	6-89
*WAI	6-89
HP-IB Message Reference	6-90
Go To Local (GTL)	6-90
Group Execute Trigger (GET)	6-90
Interface Clear (IFC)	6-90

Device Clear (DCL) or Selected Device Clear (SDC)	6-91
Local Lockout (LLO)	6-91
Remote	6-92
Serial Poll (SPOLL)	6-92
Command Quick Reference	6-93

A. Specifications

A-1	
Real Time Clock	A-1
CLK10	A-1
Trigger Input	A-1
Memory	A-1
Power Requirement	A-2
Cooling Requirements	A-2
SCPI Conformance Information	A-3
Switchbox Configuration	A-3
Multimeter Commands	A-4
Counter Commands	A-6
D/A Converter Commands	A-8
Digital I/O Commands	A-9
System Instrument Commands	A-10

B. Error Messages

Using This Appendix	B-1
Reading an Instrument's Error Queue	B-1
Error Types	B-2
Command Errors	B-2
Execution Errors	B-2
Device-Specific Errors	B-2
Query Errors	B-2
Start-up Error Messages and Warnings	B-6

C. Command Module A16 Address Space

About this Appendix	C-1
Register Addressing	C-1
The Base Address	C-2
Register Offset	C-2

D. Sending Binary Data Over RS-232

About this Appendix	D-1
Formatting Binary Data for RS-232 Transmission	D-1
Sending Binary Data Over RS-232	D-2
Setting Up the Mainframe	D-2

Product Overview

About this Chapter

This chapter contains a functional, electrical, and physical overview of the HP E1405 Command Module. These overviews are in the following sections:

- Command Module Functional Description 1-1
- Command Module Electrical Description 1-2
- Command Module Physical Description 1-2
- Command Module Memory 1-3

Command Module Functional Description

The HP E1405 Command Module is the foundation of a VXIbus system (Figure 1-1). Though its role in a VXIbus system is largely transparent (i.e., its functions need not be programmed by the user), it provides the following key functions:

- The Command Module is the Hewlett-Packard Interface Bus (HP-IB) to VXIbus interface.
- The Command Module is the Standard Commands for Programmable Instruments (SCPI) translator for Hewlett-Packard register based instruments.
- The Command Module provides the VXIbus Slot 0 and Resource Manager capabilities.
- The Command Module can drive the VXIbus TTLTRG0-7 and ECLTRG0-1 trigger lines. The module contains BNC connectors for placing an external trigger onto the selected line(s), and for routing an internal trigger to a device external to the mainframe.
- The Command Module contains an internal clock that allows you to set and read the time and date.

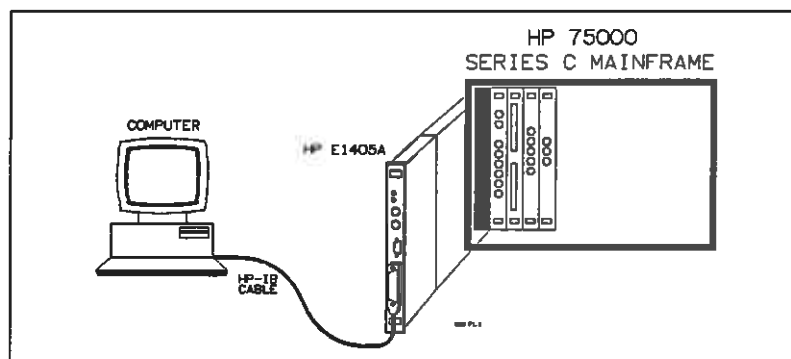


Figure 1-1. HP E1405 Based VXIbus System.

Command Module Electrical Description

The electrical characteristics of the Command Module are summarized in Appendix A: Specifications.

Command Module Physical Description

The HP E1405 Command Module occupies one C-size mainframe slot. The module faceplate has annunciators, clock and trigger connectors and interface ports that are described below.

Faceplate Annunciators

There are four annunciators on the Command Module's faceplate which show the following:

- **Failed:** shows that the Command Module has failed its power-on self test or has stopped working at some point in time.
- **SYSFAIL:** shows that the SYSFAIL line on the VXIbus backplane is being asserted by the Command Module when it fails.
- **Access:** shows that the Command Module is accessing, or being accessed by the VXIbus backplane.
- **Ready:** shows that the Command Module is in the VXIbus Normal Operation state.

Faceplate CLK10 and Trigger Connectors

There are four signal connectors on the Command Module's faceplate which function as follows:

- **Clk In:** This SMB connector allows an external 10 MHz clock to function as the system's Slot 0 CLK10 resource. This is a high impedance input with an input range from ± 50 mV to ± 5 V.
- **Clk Out:** This SMB connector allows the internal Slot 0 CLK10 resource to be routed to other VXIbus mainframes. This output drives 50 Ω .
- **Trig In:** This BNC connector allows an external trigger signal (TTL levels) to be applied to the system on the trigger line selected (TTLTRG0-7/ECLTRG0-1). The input impedance is 5 k Ω .
- **Trig Out:** This BNC connector allows an internal trigger on the trigger line specified (TTLTRG0-7/ECLTRG0-1) to be applied to an external device. This output drives 50 Ω .

The HP-IB and RS-232 Ports

The HP-IB port allows an HP-IB cable to be connected from the Command Module to a computer, or to an external disk drive. The RS-232 port can be used as a user interface, or used for peripheral control if the Command Module contains Instrument BASIC (IBASIC). The RS-232 port is a 9-pin DTE connector. Supported terminals include: HP 700/92, 700/94, 700/22, 700/43, and Wyse® WY-30™.

The Reset Button

Located beneath the HP-IB port is a reset button which is used to reconfigure your VXIbus system and return it to the power-on state.

Installing/Removing the Command Module

Once it has been configured, the Command Module is installed in the HP 75000 Series C Mainframe by lining the module up with the slot guides and then pressing the module firmly into the backplane connectors.

The Command Module is removed by pressing up on the handle with the "hp" logo, while pressing down on the handle with the "VXIbus" logo. This extracts the module from the backplane connectors allowing the module to be removed.

NOTE

The screws located above and below the handles on the Command Module can be used to secure the installed module to the mainframe.

Command Module Memory

The E1405B comes from the factory equipped with 512Kbytes of memory. An additional 512Kbytes (for a total of 1Mbyte) can be added if required.

Battery Backed Functions

The Command Module's clock and calendar functions, the user non-volatile RAM (NVRAM), and the Device Driver RAM (DDRAM) are backed up by a NiCad battery. For systems with 512Kbytes of memory this battery has a ten month lifetime and is fully recharged when the module is in the mainframe and the power has been on for fifteen continuous hours. For systems with 1Mbyte of memory this battery has a five month lifetime.

Modifying Your Configuration

About this Chapter

One purpose of the HP E1405 Command Module is to provide the resource manager function required by VXIbus systems. This chapter describes the resource manager function and shows you how to modify the resource manager's configuration process with user tables you download into non-volatile user RAM.

The main sections of this chapter include:

- System Configuration Sequence 2-1
- Identifying Statically and Dynamically Configured Modules .. 2-2
- Setting up VXI-MXI Configuration 2-7
- Setting Commander/Servant Hierarchies 2-17
- A24/A32 Address Mapping 2-22
- Interrupt Line Allocation 2-30
- Starting System Operation 2-37

System Configuration Sequence

As mentioned in the C-Size VXIbus Systems "Installation and Getting Started Guide", the resource manager (RM) within the Command Module performs the following system configuration sequence when power is applied:

- Identify all statically and dynamically configured modules
- Set commander/servant hierarchies
- Perform A24/A32 address mapping
- Allocate interrupt lines to manage communication between interrupt handler modules and interrupter modules
- Start system operation

The following sections describe each step of the configuration sequence. Included are examples on how to change the sequence using configuration tables stored in (non-volatile) user RAM.

NOTE

Refer to the C-Size VXIbus Systems "Installation and Getting Started Guide" for information on configuring the HP E1405 Command Module as the resource manager.

Identifying Statically and Dynamically Configured Modules

Statically configured modules are plug-in modules whose logical addresses are set with logical address switches. Dynamically configured modules are plug-in modules whose logical addresses are programmed (set) by the resource manager (RM).

Statically Configured (SC) Modules

Once all power-on self tests have completed, the RM identifies all statically configured (SC) modules. The RM retains information such as the module's logical address, slot number, model number, manufacturer's code, etc.

Dynamically Configured (DC) Modules

Once all SC modules have been located in a mainframe and no SC module is logical address 255, the resource manager identifies all dynamically configured (DC) modules and assigns them logical addresses as follows.

- The resource manager locates DC modules by scanning each mainframe slot. Refer to the plug-in module manual for additional information on setting up the module prior to its dynamic configuration.
- Beginning with the lowest mainframe slot (excluding slot 0), the resource manager scans each slot via the module identification (MODID) bus until a DC module is located. The module is assigned a logical address that is the lowest available multiple of 8.

The resource manager continues scanning until the next DC module is located. The module is assigned a logical address that is the next available multiple of 8. The process continues until all DC devices have been assigned logical addresses. If all multiples of 8 are used, the DC module is assigned the first available address.

- Logical addresses used by SC devices will not be assigned to DC devices.
- DC devices will not be assigned logical address 255.
- A set of address blocked DC devices will be assigned successive logical addresses beginning with the lowest available multiple of 8.

User-Defined Dynamic Configuration

If your system contains instruments comprised of multiple modules that must have successive logical addresses, then the modules must be statically configured using their logical address switches, or be dynamically configured with the user-defined dynamic configuration table. The dynamic configuration table covered in this section allows you to override the "default" configuration process by assigning logical addresses as you choose.

The Dynamic Configuration Table

User-defined dynamic configurations are specified with a dynamic configuration table created in the Command Module. The table is created as follows:

1. Table space in the Command Module's non-volatile user RAM is made available by allocating a segment of RAM with the command:
DIAGnostic:NRAM:CREate < size >
2. The location (starting address) of the table in RAM is determined with the command:
DIAGnostic:NRAM:ADDRes?
3. Data is downloaded into the table with the command:
DIAGnostic:DOWNload < address > < data >
4. The table is linked to the appropriate algorithm in the Command Module processor with the command:
VXI:CONFigure:DCTable < address >

Table Format The format of the dynamic configuration table is shown in Table 2-1.

Table 2-1. Dynamic Configuration Table Format

Valid Flag	Number of Entries		
Slot Number	Slot 0 Laddr	Laddr	Block Size
.	.	.	.
.	.	.	.
Slot Number	Slot 0 Laddr	Laddr	Block Size

The table parameters are:

Valid Flag (1/0): 1 indicates the table is valid and the modules can be configured accordingly. 0 will cause an error message (Error 39). Valid Flag is part of the table header and is one byte.

Number of Entries (1 - 254): is the number of entries in the table. Number of Modules is part of the table header and is one byte.

Slot Number (1 - 12): is the mainframe slot the module to be assigned an address is installed in. Field is one byte.

Slot 0 Laddr: is the logical address (0) of the slot 0 device in mainframe slot 0. Field is one byte.

Laddr (1-254): is the logical address to which the module in Slot Number is set. Field is one byte.

Block Size (1-128): is the number of devices in an address block. When there is more than one device, Laddr specifies the logical address of the first device in the set. The remaining devices are assigned sequential logical addresses beginning with the next highest address. When there are multiple devices in a slot that are not address blocked, there must be an entry in the table for each device. Field is one byte.

Determining the Table Size

The dynamic configuration table has a two byte header and each of the four fields are one byte. The amount of RAM to allocate with DIAG:NRAM:CRE is computed as:

$$2 + 4(N)$$

where N is the number of modules to be configured. For example, to dynamically configure three modules based on logical addresses you've selected, the table size would be: $2 + 4(3) = 14$ bytes. DIAG:NRAM:CRE would be executed as:

```
OUTPUT @E1405;"DIAG:NRAM:CRE 14"
```

Data Format

Data must be sent to the dynamic configuration table in binary bytes. This is accomplished by reading the data into an Integer variable in the computer, and then downloading the data to the table using the ANSI/IEEE 488.2-1987 Arbitrary Block Program Data format. This process is shown in the example program which follows. More information on the Arbitrary Block Program format can be found in Chapter 5, and in the ANSI/IEEE 488.2-1987 document.

CAUTION

When downloading data into the dynamic configuration table, DIAGnostic:DOWNload does not determine if the table is large enough to store the data. If the amount of data sent by DIAGnostic:DOWNload is greater than the (table) space allocated by DIAGnostic:NRAM:CREate, system errors will occur. You can recover from these errors by executing DIAG:BOOT:COLD, or by pressing "Ctrl R" on an RS-232 terminal while cycling mainframe power.

Example: Dynamically Configuring a Module

The following program dynamically sets the logical address of the HP E1410 6 1/2-Digit Multimeter to 32. The program notes each of the steps used to create and load the table.

To dynamically configure the multimeter, its logical address must be set to 255 using the logical address switches.

NOTE

The computer syntax shown is for an external controller or for an embedded (HP V/360) controller.

```
10  !Assign an I/O path and allocate a variable to store dynamic
20  !configuration data to be downloaded to the command module.
30  ASSIGN @E1405 TO 70900;EOL CHR$(10) END
40  INTEGER Dy_config(1:6)
50  !
60  !Allocate a segment of non-volatile user RAM on the Command
70  !Module to store the dynamic configuration table (1 module).
80  OUTPUT @E1405;"DIAG:NRAM:CRE 6"
90  !
100 !Re-start the system instrument to allocate the user RAM. Wait for the
110 !re-start to complete before continuing.
120 OUTPUT @E1405;"DIAG:BOOT:WARM"
130 ON TIMEOUT 7,.1 GOTO Complete
140 Complete:  B = SPOLL(70900)
150 OFF TIMEOUT 7
160 !
170 !Return the starting address of the table in non-volatile user RAM.
180 OUTPUT @E1405;"DIAG:NRAM:ADDR?"
190 ENTER @E1405;A
200 !
210 !Download the following bytes: the table is valid, one module is
220 !dynamically configured, it's installed in slot 6, the logical address of
230 !the slot 0 module is 0, the logical address to be set is 32, and the
240 !block size is 1.
250 DATA 1,1,6,0,32,1
260 READ Dy_config(*)
270 OUTPUT @E1405 USING "#,3(K)";"DIAG:DOWN ";A;" ,#0"
280 OUTPUT @E1405 USING "B";Dy_config(*)
290 !
300 !Link the dynamic configuration table to the appropriate algorithm.
310 OUTPUT @E1405;"VXI:CONF:DCT ";A
320 !
330 !Re-start the system instrument to set the user-defined configuration.
340 OUTPUT @E1405;"DIAG:BOOT:WARM"
350 END
```

Comments

- You can verify the dynamic configuration using the System Verification program found in the C-Size VXIbus Systems "Installation and Getting Started Guide".

- Errors associated with dynamic configurations are:

ERROR 1: FAILED DEVICE

This error occurs when a dynamically configured device at logical address 255 failed during its power on sequence.

ERROR 4: DC DEVICE ADDRESS BLOCK TOO BIG

This error occurs when the block size specified in the table is greater than 127.

ERROR 7: DC DEVICE MOVE FAILED

This error occurs when a device was not set to the logical address specified, possibly due to a hardware failure on the module. The error also occurs when all devices in an address block did not move.

ERROR 9: UNABLE TO MOVE DC DEVICE

This error occurs when there are not enough successive logical addresses available for the specified block size, or if the logical address specified is already occupied by another static or dynamic module.

ERROR 39: INVALID UDEF DC TABLE

This error occurs when the table's valid flag does not equal 1.

ERROR 40: INVALID UDEF DC TABLE DATA

This error occurs when there are greater than 254 entries in the table.

- The logical addresses assigned by the dynamic configuration table are used by the system until DIAGnostic:BOOT:COLD is executed, or until VXI:CONFigure:DCTable 0 is executed.

Setting Up VXI-MXI Configuration

During configuration, if an extender device is present the resource manager will attempt to assign logical addresses and memory according to the rules listed below. You can override these rules by creating a user defined Extender table.

Logical Address Configuration

The following rules and recommendations apply to assigning logical addresses. For a more detailed discussion of how to lay out logical addresses please refer to the *VXI-MXI Bus Extender User's Manual*.

- The window of a child side (local) extender must include the logical addresses of all parent side extenders on its interconnect bus.
- The downward window of a child side extender cannot include any devices which are not its descendants, except its own address. It must include all devices on all of its own descendant busses.
- A child side extender should have a higher logical address than any SC or DC devices on its VME bus (excluding other child side extenders).
- A child side extender should have a lower logical address than any of its corresponding parent side extenders and stand alone devices on its interconnect bus.
- A parent side (remote) extender should have the lowest logical address on its own VME bus.
- The logical address of a parent side extender can be lower than the address of its corresponding child side extender on its interconnect bus.

Default Logical Address Assignments

The resource manager will attempt to assign logical addresses to dynamically configured devices according to the following rules:

- The window for a child side extender will be set outward to the minimum possible size to include all of the logical addresses found on all of its descendant busses. This includes all stand alone devices and all parent side extenders that are descendants of the child side extender.

NOTE

The window for a child side extender may or may not include the logical address of the child side extender itself.

- The window for a parent side extender will be set inward to the minimum possible size to include all of the devices on its VME bus and all of its descendants.

NOTE

The window for a parent side extender may or may not include the logical address of the parent side extender itself.

- A Dynamically configured (DC) device will be assigned a logical address as follows:
 - DC devices on a given VME bus will be assigned logical addresses after all descendant busses of that VME bus have been configured.
 - DC devices on a given VME bus will be assigned addresses in the range defined by the Static (SC) device with the lowest logical address on that VME bus and the maximum allowable logical address for that VME bus.
 - Each DC device will be assigned an address that is a multiple of 8 within the allowable range for that VME bus until all of these addresses have been used.
 - Any additional DC devices will be assigned the lowest available addresses within the allowable range for that VME bus.

A16/A24/A32 Address Window Configuration

The following rules and recommendations apply to assigning A16/A24/A32 memory addresses. For a more detailed discussion of how to lay out memory addresses please refer to the *VXI-MXI Bus Extender User's Manual*.

- Systems with multiple VME devices should be configured so that the VME devices in mainframes whose parent side extenders have the highest logical addresses should also have the highest memory addresses.
- VME devices should be configured to have the lowest addresses on their particular VME bus.

Default A16/A24/A32 Address Window Assignments

The resource manager will not attempt to perform any A16 address window configuration as a default. It will attempt to configure A24 and A32 memory according to the following rules:

- A memory page is 1/256 of the total memory space. The minimum size of an A24 or A32 memory window is 2 pages and the maximum size of the window is 256 pages as defined in VXI-6. For A24 memory a single page is 65,536 bytes and the minimum window size is 131,072 bytes. For A32 memory a single page is 16,777,216 bytes and the minimum window is 33,554,432 bytes.
- The base address of a memory window must be zero or an even multiple of the size of the window.
- The window for a child side extender will be set to the minimum possible size to include all of the memory addresses found on all of its descendants.
- The window for a parent side extender will be set to the minimum possible size to include all of the memory on its VME bus and all of its descendants.

- A VXI device will be assigned a memory location in the following manner:
 - VXI devices on a given VME bus will be assigned memory locations after all descendant busses of the VME bus have been configured.
 - VXI bus devices on a given VME bus will be assigned memory locations in the range defined by the lowest and highest memory pages available for that bus.
 - The first available page for a VME bus will be the first page that is higher than any reserved page on any of its ancestors.
 - VXI bus devices will be assigned the lowest memory locations available on the current bus
 - VXI devices will be assigned locations according to memory size and logical address in that order. The device with the largest memory size on a given bus will be assigned an address first. For devices with the same size, the device with the lowest logical address will be assigned a memory location first.
 - If possible, no devices will be assigned to memory locations in the bottom or top 1/8 of the total memory (e.g., in A24 memory addresses 000000_{16} - 200000_{16} or $E00000_{16}$ - $FFFFFF_{16}$).
- VME reserved memory must be placed in locations that will not interfere with windows previously configured. The only way the resource manager can know the location(s) of VME memory is for you to provide this information in the user defined Memory table (see *A24/A32 Address Mapping* later in this chapter for more details).

Interrupt Register Configuration

The rules listed below will be used to assign the configuration of the INTX interrupt register during system startup unless you override them with entries in the user defined Extender table.

- The interrupt enable bits in the INTX interrupt configuration register on every extender will be enabled for each VME interrupt line that has a VXI handler assigned.
- The interrupt enable bits in the INTX interrupt register on every extender will be disabled for each VME interrupt line that has no VXI handler assigned.
- For every VME interrupt line that has a VXI interrupt handler assigned, the direction will be set on each extender such that an interrupt on that line will be routed towards the VME backplane that contains the handler.

TTL Trigger Register Configuration

The TTL Trigger Register will be set to $C0C0_{16}$ (TTL triggers disabled) for all parent side and child side extenders that support TTL triggers. you may enable TTL triggers and set the TTL trigger directions with the Extender table.

ECL Trigger Register Configuration

The ECL Trigger Register will be set to C0C016 (ECL triggers disabled) for all parent side and child side extenders that support ECL triggers. you may enable ECL triggers and set the ECL trigger directions with the Extender table.

Utility Register Configuration

The default utility register configuration is shown in table 2-2. Since the resource manager may have to reboot during the system configuration process, (e.g., to download a driver), the utility register is not a part of the Extender table. This will help ensure that that the SYSRESET signal will propagate throughout the system during a reboot so that all of the cards will receive a hard reset.

If you wish to alter the contents of the Utility Register you can use DIAG:POKE commands directly to the registers. Keep in mind that this may alter the default system reboot process.

Table 2-2. Utility Register Default Configuration

Extender Type	ACFIN	ACFOUT	SFIN	SFOUT	SRIN	SROUT
Child Side	enabled	enabled	enabled	enabled	enabled	enabled
	(1)	(1)	(1)	(1)	(1)	(1)
Parent Side	enabled	enabled	enabled	enabled	enabled	enabled
	(1)	(1)	(1)	(1)	(1)	(1)

User Defined Logical Address and Memory Windows

In many systems that use extenders, the standard bootup algorithms will not be suitable for your configuration. In such systems it will be necessary to unambiguously define your logical address and memory mapping for the bootup configuration routine.

The User Defined Extender Table

You can define your own logical address and memory mapping in a system with extenders by using the user defined Extender table. This table is created as follows:

1. Table space in the Command Module's non-volatile user RAM is made available by allocating a segment of RAM with the command:
DIAGnostic:NRAM:CREate <size >
2. The location (starting address) of the table in RAM is determined with the command:
DIAGnostic:NRAM:ADDRes?
3. Data is downloaded into the table with the command:
DIAGnostic:DOWNload < address > < data >
4. The table is linked to the appropriate algorithm in the Command Module processor with the command:
VXI:CONFigure:ETABle < address >

Table Format The user defined Extender table consists of a two byte header followed by the required number of extender records. The first byte of the header is a Table Valid flag (1 = valid) and the second byte specifies the number of records in the table.

valid flag (0 1)
of records (N)
extender record 1
extender record 2
•
•
extender record N

Any single item in an extender record can be disabled so that the resource manager will perform the default configuration for the item. For example, to use the resource manager default algorithm for interrupt enable, set the appropriate field in the extender record (see table 2-3) to 255.

Table2-3. User Defined Extender Table Record

Field	Description	Format	Range	Field Disable Value
1	Logical Address (parent or child side extender)	int16	1-255	n/a
2	Logical Address Window Base	int16	0-254 ¹	255
3	Logical Address Window Size	int16	2-256	n/a
4	A16 Memory Base Page	int16	0-254 ¹	255
5	A16 Memory Window Size (number of pages)	int16	2-256	n/a
6	A24 Memory Base Page	int16	0-254 ¹	255
7	A24 Memory Window Size (number of pages)	int16	2-256	n/a
8	A32 Memory Base Page	int16	0-254 ¹	255
9	A32 Memory Window Size (number of pages)	int16	2-256	n/a
10	Interrupt Enable	int16	n/a ²	255
11	TTL Trigger enable	int16	n/a ³	255
12	ECL Trigger Enable	int16	n/a ⁴	255

1 The upper byte of this field (bits 15-8) is reserved.

2 This is Mainframe Extender Register 12₁₆. See the VXI-6 specification or your Mainframe Extender manual for a definition of this register. Interrupts may not be supported by all Mainframe Extender cards.

3 This is Mainframe Extender Register 14₁₆. See the VXI-6 specification or your Mainframe Extender manual for a definition of this register. TTL triggers may not be supported by all Mainframe Extender cards.

4 This is Mainframe Extender Register 16₁₆. See the VXI-6 specification or your Mainframe Extender manual for a definition of this register. ECL triggers may not be supported by all Mainframe Extender cards.

Determining the Table Size

The user defined Extender table has a one word header and each of the 12 fields is also one word. The amount of RAM allocated with DIAG:NRAM:CRE is specified in bytes. Since one word is two bytes, the amount of RAM to allocate is computed as:

$$2 + 24(N)$$

where N is the number of modules to be configured. For example, to provide information for three extender devices, the table size would be:

$$2 + 24(3) = 74 \text{ bytes.}$$

DIAG:NRAM:CRE would be executed as:

```
OUTPUT @E1405;"DIAG:NRAM:CRE 74"
```

Data Format

Data must be sent to the Extender table in 16-bit words. This is accomplished by reading the data into an Integer variable in the computer, and then downloading the data to the table using the ANSI/IEEE 488.2-1987 Arbitrary Block Program Data format. This process is shown in the example program which follows. More information on the Arbitrary Block Program format can be found in Chapter 5, and in the ANSI/IEEE 488.2-1987 document.

The table header is sent as a single 16-bit word which must contain the Valid Flag and the number of modules involved. For a valid table, the header is 256 plus the number of modules. For example, to indicate a valid table with seven entries, the header is 263 (256 + 7 = 263).

CAUTION

When downloading data into the user defined Extender table, DIAGnostic:DOWNload does not determine if the table is large enough to store the data. If the amount of data sent by DIAGnostic:DOWNload is greater than the table space allocated by DIAGnostic:NRAM:CREate, system errors will occur. You can recover from these errors by executing DIAG:BOOT:COLD, or by pressing "Ctrl R" on an RS-232 terminal which cycling mainframe power.

**Example: User Defined
Extender Table**

This example shows a single interconnect bus with a child side (local) extender at logical address 63 in the root mainframe and a parent side (remote) extender at logical address 64 in the secondary mainframe.

258	valid (upper byte) + 2 records (lower byte)
63	child side extender logical address
128	logical address window base
64	logical address window size (128 to 191)
255	Specify no A16 memory
0	A16 memory size (ignored)
64	A24 memory base page
64	A24 memory size (pages 64 to 127)
0	A32 memory base page
128	A32 memory size (pages 0 to 127)
257	interrupt line 1 enabled (IN)
769	TTL triggers (TTL1, OUT, TTL0 IN)
-15936	ECL triggers (C1C0 ₁₆ = ECL0 enabled OUT)
64	parent side extender logical address
128	logical address window base
64	logical address window size (128 to 191)
255	Specify no A16 memory
0	A16 memory size (ignored)
64	A24 memory base page
64	A24 memory size (pages 64 to 127)
255	Specify no A32 memory
0	A32 memory size (ignored)
256	interrupt line 1 enabled (OUT)
770	TTL triggers (TTL1 IN, TTL0 OUT)
-15935	ECL triggers (ECL0 IN)

The following program downloads the table shown above into user non-volatile memory. The program notes each of the steps used to create and load the table.

NOTE

The computer syntax shown is for an external controller or for an embedded (HP V/360) controller.

```

10 !Assign an I/O path and allocate a variable to store MXI
20 !configuration data to be downloaded to the command module.
30 ASSIGN @E1405 TO 70900;EOL CHR$(10) END
40 INTEGER MXI_config(1:25)
50 !
60 !Allocate a segment of non-volatile user RAM on the Command
70 !Module to store the user defined MXI table (1 module).
80 OUTPUT @E1405;"DIAG:NRAM:CRE 50"
90 !
100 !Re-start the system instrument to allocate the user RAM. Wait for the
110 !re-start to complete before continuing.
120 OUTPUT @E1405;"DIAG:BOOT:WARM"
130 ON TIMEOUT 7,.1 GOTO Complete
140 Complete: B = SPOLL(70900)
150 OFF TIMEOUT 7
160 !
170 !Return the starting address of the table in non-volatile user RAM.
180 OUTPUT @E1405;"DIAG:NRAM:ADDR?"
190 ENTER @E1405;A
200 !
210 !Download the required bytes.
220 !see table above for the meaning of these bytes
250 DATA 258, 63, 128, 64, 0, 0, 64, 64, 0, 128, 257, 769, -15936, 64,
128, 64, 0, 0, 64, 16, 0, 0, 256, 770, -15935
260 READ MXI_config(*)
270 OUTPUT @E1405 USING "#,3(K)";"DIAG:DOWN ";A;" ,#0"
280 OUTPUT @E1405 USING "W";MXI_config(*)
290 !
300 !Link the user defined MXI table to the appropriate algorithm.
310 OUTPUT @E1405;"VXI:CONF:ETAB ";A
320 !
330 !Re-start the system instrument to set the user-defined configuration.
340 OUTPUT @E1405;"DIAG:BOOT:WARM"
350 END

```

Comments

- The following errors are associated with the Extender table or indicate that you may need to create an Extender table.

ERROR 50: EXTENDER NOT SLOT 0 DEVICE

This error occurs when a parent side VXIbus extender in a remote mainframe is not in slot 0 of its mainframe. The resource manager expects all parent side VXIbus extenders to be installed in slot 0 of their mainframe.

ERROR 51: INVALID EXTENDER LADD WINDOW

This error occurs when the configuration routine finds an invalid start address or size for an extender logical address window. You should reconfigure the logical addresses of the VXI devices or create a user defined Extender table for the system to override the default algorithm.

ERROR 52: DEVICE OUTSIDE OF LADD WINDOW

This error occurs when a device or devices were found outside the default maximum or outside the user defined range for the extender. You should reconfigure the logical addresses of the VXI devices or create a new Extender table for the system to override the default algorithm.

ERROR 53: INVALID EXTENDER A24 WINDOW

This error occurs when the configuration routine finds an invalid start address or size for an extender A24 address window. You should reconfigure the VME memory devices or create a user defined Extender table to override the default algorithm.

ERROR 54: DEVICE OUTSIDE OF A24 WINDOW

This error occurs when an A24 memory device is located outside of the allowable logical address range of an MXIbus extender. You should reconfigure the VME memory devices or create a user defined Extender table to override the default algorithm.

ERROR 55: INVALID EXTENDER A32 WINDOW

This error occurs when the configuration routine finds an invalid start address or size for an extender A32 address window. You should reconfigure the VME memory devices or create a user defined Extender table to override the default algorithm.

ERROR 56: DEVICE OUTSIDE OF A32 WINDOW

This error occurs when an A32 memory device is located outside of the allowable logical address range of a MXIbus extender. You should reconfigure the VME memory devices or create a user defined Extender table to override the default algorithm.

ERROR 57: INVALID UDEF LADD WINDOW

This error occurs when a user defined logical address window violates the VXI-6 specification (has an invalid base or size). You should redefine your Extender table with correct values.

ERROR 58: INVALID UDEF A16 WINDOW

This error occurs when a user defined A16 window violates the VXI-6 specification (has an invalid base or size). You should redefine your Extender table with correct values.

ERROR 59: INVALID UDEF A24 WINDOW

This error occurs when a user defined A24 window violates the VXI-6 specification (has an invalid base or size). You should redefine your Extender table with correct values.

ERROR 60: INVALID UDEF A32 WINDOW

This error occurs when a user defined A32 window violates the VXI-6 specification (has an invalid base or size). You should redefine your Extender table with correct values.

ERROR 61 INVALID UDEF EXT TABLE

This error occurs when the valid flag is not set to 1 in the Extender table. You should redefine your Extender table with correct values.

ERROR 62: INVALID UDEF EXT TABLE DATA

This error occurs when there is an incorrect number of records for a user defined Extender table. You should make sure that the number of records shown in the header matches the number of records actually in the table.

ERROR 63: UNSUPPORTED UDEF TTL TRIGGER

This error occurs when there is a user defined Extender table TTL trigger entry for a MXIbus extender that does not support TTL triggers.

ERROR 64: UNSUPPORTED UDEF ECL TRIGGER

This error occurs when there is a user defined Extender table ECL trigger entry for a MXIbus extender that does not support ECL triggers.

ERROR 66: INTX CARD NOT INSTALLED

This error occurs when the INTX card is not installed on the VXI-MXI extender. You should make sure that the INTX card is correctly installed and that it is functioning.

- The system configuration assigned by the Extended Device table is used by the system until DIAGnostic:BOOT:COLD is executed, or until VXI:CONFigure:ETABLE 0 is executed.

Setting Commander/Servant Hierarchies

In a VXIbus system, a commander is a plug-in module which controls other plug-in modules. "Control" can be a commander such as the HP E1405 Command Module translating SCPI commands, and/or serving as the HP-IB interface for (servant) modules within its servant area.

During the configuration sequence, the resource manager assigns servant modules to a commander module based on the servants' logical addresses and the commander's servant area. The concept of the servant area is shown in Figure 2-1. The C-Size VXIbus Systems "Installation and Getting Started Guide" shows how to set the Command Module's servant area.

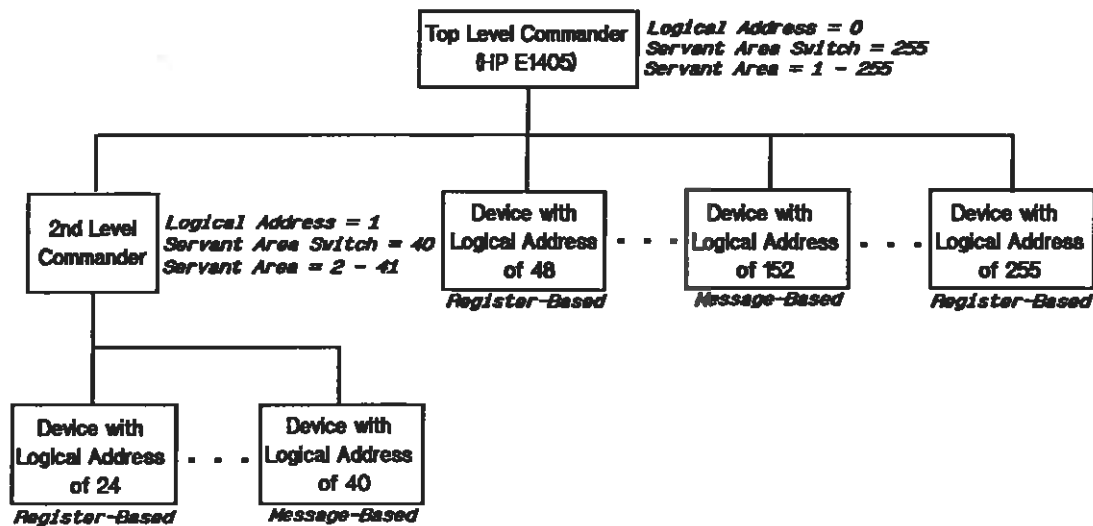


Figure 2-1. Example Commander/Servant Hierarchy

Note the following regarding commander/servant relationships.

- A commander's servant area is its logical address + 1, through its logical address + its servant area switch setting.
- If within a given commander's servant area (Figure 2-1) there is another lower-level commander(s) (logical address 1), the given commander will control the lower-level commander. However, all modules within the servant area of the lower-level commander (logical addresses 2 - 41) will be controlled by the lower-level commander.
- If there is a commander outside the servant area of the Command Module/resource manager, that commander becomes a top level commander. The resource manager will assign all modules within the commander's servant area to that commander, or to that commander's lower-level commanders.
- The Command Module will always be the commander for IBASIC even if IBASIC's logical address (240) is outside the module's servant area. There can be multiple IBASICs in the same system since each is a servant to its respective Command Module. Note that there are no VXIbus registers for IBASIC.

User-Defined Commander/Servant Hierarchies

In some systems you may need to assign a servant to a commander that is outside the commander's servant area. In other systems, it may be necessary to change a module's secondary HP-IB address, or assign secondary addresses to modules whose logical addresses are not instrument identifiers. These tasks can be accomplished with the user-defined Commander/Servant Hierarchy table described in this section.

NOTE

Register based instrument drivers that support multiple card sets normally require that the cards in the set have sequential logical addresses (see "How to Create and Instrument" in the *C-Size VCXIBus System Installation and Getting Started Guide*). When instrument drivers support non-sequential logical addresses, instruments that consist of non-sequential card sets must be created using the user defined Commander/Servant Hierarchy table. There must be an entry in the table for every card in the instrument card set.

The Commander/Servant Hierarchy Table

User defined commander/servant hierarchies and secondary HP-IB addresses are specified with a Commander/Servant Hierarchy table created in the Command Module. The table is created as follows:

1. Table space in the Command Module's non-volatile user RAM is made available by allocating a segment of RAM with the command:

DIAGnostic:NRAM:CREate < size >

2. The location (starting address) of the table in RAM is determined with the command:

DIAGnostic:NRAM:ADDRes?

3. Data is downloaded into the table with the command:

DIAGnostic:DOWNload < address > < data >

4. The table is linked to the appropriate algorithm in the Command Module processor with the command:

VXI:CONFigure:CTABLE < address >

Table Format

The format of the commander/servant hierarchy table is shown in Table 2-4.

Table 2-4. Commander/Servant Hierarchy Table Format

Valid Flag/ Number of Modules		
Laddr	Cmdr Laddr	Sec Addr
Laddr	Cmdr Laddr	Sec Addr
.	.	.
Laddr	Cmdr Laddr	Sec Addr

The table parameters are:

- **Valid Flag (1/0):** 1 indicates the table is valid and the modules should be configured accordingly. 0 will cause an error message (Error 38). Valid Flag is part of the table header and is represented by the upper eight bits of the header word.

- **Number of Modules (1 - 254):** is the number of entries in the table. Number of Modules is part of the table header and is represented by the lower eight bits of the header word.
- **Laddr:** is the logical address of the module which is assigned a new commander or new secondary HP-IB address. Field is one word.
- **Cmdr Laddr:** is the logical address of the commander to which the module specified by Laddr is assigned. If -1 is specified, the module is not assigned to a commander. Field is one word.
- **Sec Addr (1 - 30):** is the secondary HP-IB address assigned to the module specified by Laddr. If -1 is specified, the secondary address is assigned by default (see "How to Create an Instrument" in the C-Size VXibus Systems "Installation and Getting Started Guide"). Field is one word.

Determining the Table Size

The commander/servant hierarchy table has a one word header and each of the three fields is also one word. The amount of RAM allocated with DIAG:NRAM:CRE is specified in bytes. Since one word is two bytes, the amount of RAM to allocate is computed as:

$$2 + 6(N)$$

where N is the number of modules to be configured. For example, to assign three modules to a particular commander, the table size would be:
 $2 + 6(3) = 20$ bytes. DIAG:NRAM:CRE would be executed as:

```
OUTPUT @E1405;"DIAG:NRAM:CRE 20"
```

Data Format

Data must be sent to the commander/servant hierarchy table in 16-bit words. This is accomplished by reading the data into an Integer variable in the computer, and then downloading the data to the table using the ANSI/IEEE 488.2-1987 Arbitrary Block Program Data format. This process is shown in the example program which follows. More information on the Arbitrary Block Program format can be found in Chapter 5, and in the ANSI/IEEE 488.2-1987 document.

The table header is sent as a single 16-bit word which must contain the Valid Flag and the number of modules involved. For a valid table, the header is 256 plus the number of modules. For example, to indicate a valid table with seven entries, the header is 263 ($256 + 7 = 263$).

CAUTION

When downloading data into the commander/servant hierarchy table, DIAGnostic:DOWNload does not determine if the table is large enough to store the data. If the amount of data sent by DIAGnostic:DOWNload is greater than the (table) space allocated by DIAGnostic:NRAM:CREate, system errors will occur. You can recover from these errors by executing DIAG:BOOT:COLD, or by pressing "Ctrl R" on an RS-232 terminal which cycling mainframe power.

Example: Assigning a Secondary HP-IB Address

The following program assigns secondary HP-IB address 01 to the HP E1411 5 1/2-Digit Multimeter at logical address 25. The program notes each of the steps used to create and load the table.

NOTE

The computer syntax shown is for an external controller or for an embedded (HP V/360) controller.

```
10 !Assign an I/O path and allocate a variable to store commander/
20 !servant hierarchy data to be downloaded to the command
   module.
30 ASSIGN @E1405 TO 70900;EOL CHR$(10) END
40 INTEGER Cs_hier(1:4)
50 !
60 !Allocate a segment of non-volatile user RAM on the Command
70 !Module to store the commander/servant hierarchy table.
80 OUTPUT @E1405;"DIAG:NRAM:CRE 8"
90 !
100 !Re-start the system instrument to allocate the user RAM. Wait for the
110 !re-start to complete before continuing.
120 OUTPUT @E1405;"DIAG:BOOT"
130 ON TIMEOUT 7,.1 GOTO Complete
140 Complete: B = SPOLL(70900)
150 OFF TIMEOUT 7
160 !
170 !Return the starting address of the table in non-volatile user RAM.
180 OUTPUT @E1405;"DIAG:NRAM:ADDR?"
190 ENTER @E1405;A
200 !
210 !Download the following: the table is valid and one module is being
220 !assigned a secondary address, the logical address of the module is
   25,
230 !its commander's logical address is 0, the secondary address is 01.
250 DATA 257,25,0,1
260 READ Cs_hier(*)
270 OUTPUT @E1405 USING "#,3(K)";"DIAG:DOWN ";A;" ,#0"
280 OUTPUT @E1405 USING "W";Cs_hier(*)
290 !
300 !Link the commander/servant hierarchy table to the appropriate
   algorithm
320 OUTPUT @E1405;"VXI:CONF:CTAB ";A
330 !
340 !Re-start the system instrument to set the user-defined configuration.
350 OUTPUT @E1405;"DIAG:BOOT"
360 END
```


Comments

- You can verify the secondary address assigned using the System Verification program contained in the C-Size VXIbus Systems "Installation and Getting Started Guide".
- The following errors are associated with the commander/servant hierarchy table:

ERROR 12: INVALID UDEF COMMANDER LADD

This error occurs when the commander logical address specified in the table (Cmdr Laddr) is not the address of a valid commander.

ERROR 14: INVALID UDEF SECONDARY ADDRESS

This error occurs when the user defined secondary address (Sec Addr) is invalid. Valid secondary addresses are -1, 1 - 30. The error also occurs if the device to which the secondary address is assigned is outside the servant area of the Command Module.

ERROR 15: DUPLICATE SECONDARY ADDRESS

This error occurs when a secondary address is specified more than once in the same commander/servant hierarchy table.

ERROR 18: INVALID COMMANDER LADD

This error occurs when a device requiring a commander is given to a commander with logical address -1 (no commander).

ERROR 37: INVALID UDEF CNFG TABLE

This error occurs when the table's valid flag is not set to 1.

ERROR 38: INVALID UDEF CNFG TABLE DATA

This error occurs when there are greater than 254 entries in the table.

- The secondary HP-IB addresses (and/or commanders) assigned by the commander/servant hierarchy table are used by the system until DIAGnostic:BOOT:COLD is executed, or until VXI:CONFigure:CTABLE 0 is executed.

A24/A32 Address Mapping

During the configuration sequence, the resource manager reads each VXI device's ID register to determine if the device requires a block of A24 or A32 addresses. Figure 2-2 shows the address mapping concept.

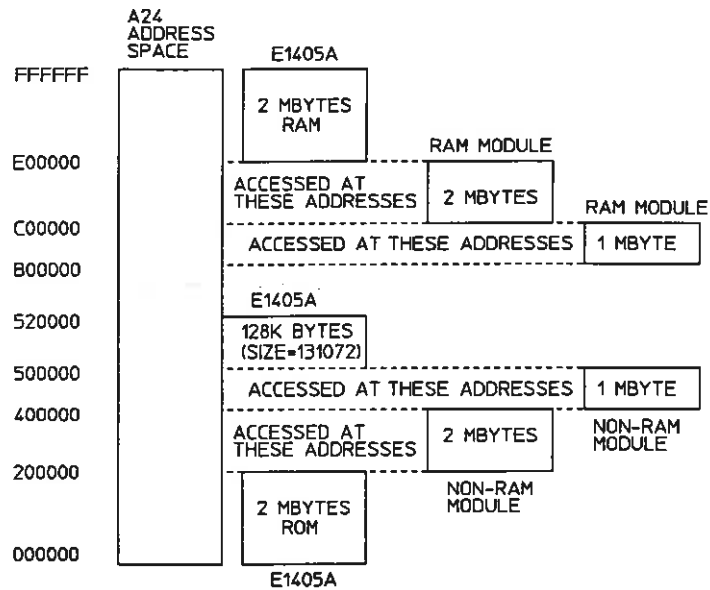


Figure 2-2. A24/A32 Address Mapping Concept (HE10) E1405-F.2.11

Address Allocation

Referring to Figure 2-2, the resource manager allocates A24 and A32 addresses as follows:

- The top and bottom 2 MBytes of A24 addresses are used by the Command Module for its own RAM and ROM.
- VXIbus modules are allocated addresses from the bottom of the address space up.
- The order of address allocation is based on the number of addresses required (memory size) and the logical address. Modules with the largest amount of memory are allocated addresses first. Modules with the same amount of memory are allocated addresses beginning with the lowest logical address.
- The top 2 Mbytes of A24 addresses (used internally by the Command Module RAM) can be allocated. However, the Command Module cannot access those addresses on the other device.
- An address allocation table can be used to reserve blocks of A24/A32 addresses for VME devices. This table is also used to assign addresses other than the default addresses assigned by the resource manager.
- The A24 address space is 16 MBytes and the A32 address space is 4 GBytes. The Command Module does not have A32 address lines and cannot access A32 address space. However, it will allocate A32 address

space for those devices which can access it. A32 memory allocation is similar to A24 memory allocation.

- A32 address space is 00000000₁₆ through FFFFFFFF₁₆.

Allocating Address Space for VME Devices

The (Command Module) resource manager has no way to determine when VME devices have been installed in the system. As a result, the resource manager allocates addresses to VXI A24/A32 devices rather than to VME devices.

There are two ways to prevent addresses intended for a VME device from being assigned to VXI devices. The first method is described below. The second method uses an address allocation table to "reserve" a block of addresses. The table used for this is described in the section "Reserving A24 Address Space".

Allocating Address Space for VME Devices: Method 1

1. Configure and install all modules (except VME devices) in the HP 75000 Series C Mainframe. This process is described in the "C-Size VXIbus Systems Installation and Getting Started Guide".
2. Turn on the mainframe and note section 4 of the resource manager's configuration sequence (Figure 2-3).

Given the starting (offset) A24 addresses assigned to the devices and the size of each device's memory (converted to hexadecimal), the A24 addresses not allocated can be determined. For example, in Figure 2-3, the highest offset is 240000₁₆ with a size of 20000₁₆ (131072 bytes converted to hexadecimal). Thus, for this system, A24 addresses from 260000₁₆ to DFFFFFF₁₆ are available to VME devices.

Note

In systems that include VXI-MXI extenders you should use a table to tell the resource manager where your A24/A32 VME memory is located. The resource manager cannot find VME memory without this table.

1	<p>Testing ROM Testing 512K Bytes RAM Passed Testing CPU CPU Self Test Passed Nonvolatile Ram Contents Lost HP-IB address: 09 Talk/Listen Command Module ladd = 0 Command Module servant area = 255</p>	<p>The HP E1405 operating system performs a series of self-tests and clears its volatile RAM. The Command Module's HP-IB address, logical address, and servant area (based on the switch settings) are reported.</p>
2	<p>Command Module VME bus timeout -- ENABLED</p>	<p>The resource manager identifies the status of the command module VME bus timeout. This must be ENABLED for systems without VXibus extenders (1405B Command Module HP-IB switch #5 = 0)</p>
3	<p>Searching for static devices in mainframe 0 SC device at ladd 0 in slot 0 SC device at ladd 8 in slot ? SC device in ladd 16 in slot 8 Searching for dynamic devices in mainframe 0 DC device in slot 3 moved to ladd 24, block size = 1</p>	<p>The resource manager identifies all statically configured modules, and then locates and configures all dynamically configurable modules.</p>
4	<p>Searching for pseudo devices</p>	<p>Pseudo devices are instruments such as IBASIC.</p>
5	<p>Configuring Commander/Servant hierarchy ladd = 0, cmdr ladd = -1 ladd = 8, cmdr ladd = 0 ladd = 16, cmdr ladd = 0 ladd = 24, cmdr ladd = 0 ladd = 32, cmdr ladd = 24 ladd = 64, cmdr ladd = 24 Validating Commander / Servant hierarchy Commander ladd 24 granted device ladd 32 Commander ladd 24 granted device ladd 64</p>	<p>The resource manager establishes the VXibus system's commander/servant hierarchies based on the commander's servant area and the servant's logical address.</p>
6	<p>Mapping A24 Memory ladd 0, offset = 00200000H, size = 131072 (bytes) ladd 24, offset = 00220000H, size = 131072 (bytes) ladd 64, offset = 00240000H, size = 131072 (bytes) Mapping A32 memory in mainframe 0</p>	<p>The resource manager allocates A24 addresses to access the memory located on the modules at logical addresses 0, 24, and 64. The offset is specified in hexadecimal and the size is specified in bytes. In this system, there are no A32 devices.</p>
7	<p>Configuring VME interrupts VME interrupt line 1 assigned to ladd 0, handler ID 1 VME interrupt line 2 assigned to ladd 24, handler ID 1 VME interrupt line 3 assigned to ladd 64, handler ID 1 VME interrupt line 4 - no handler assigned VME interrupt line 5 - no handler assigned VME interrupt line 6 - no handler assigned VME interrupt line 7 - no handler assigned</p>	<p>The resource manager allocates interrupt lines to itself and to the other interrupt handlers in the system.</p>
8	<p>SYSTEM INSTALLED AT SECONDARY ADDR 0 VOLTMR INSTALLED AT SECONDARY ADDR 1 SWITCH INSTALLED AT SECONDARY ADDR 2 Mbinstr INSTALLED AT SECONDARY ADDR 3 SYSTEM instrument started BNO issued to ladd 24, BNO response = FFFE Opening HP-IB access for message based device at sec addr 03</p>	<p>The resource manager identifies the secondary HP-IB addresses used in the system, starts the system instrument (i.e. Command Module), issues the Begin Normal Operation (BNO) command to its direct message based servant, and opens HP-IB access to the module at secondary HP-IB address 03.</p>

Figure 2-3. RM Configuration Without Extenders

1	<p>Testing ROM Testing 512K Bytes RAM Passed Testing CPU CPU Self Test Passed Nonvolatile Ram Contents Lost HP-IB address: 09 Talk/Listen Command Module ladd = 0 Command Module servant area = 255</p>	<p>The HP E1405 operating system performs a series of self-tests and clears its volatile RAM. The Command Module's HP-IB address, logical address, and servant area (based on the switch settings) are reported.</p>
2	<p>Command Module VME bus timeout -- DISABLED</p>	<p>The resource manager identifies the status of the command module VME bus timeout. This must be DISABLED for systems without VXibus extenders (1405B Command Module HP-IB switch #5 = 0)</p>
3	<p>Searching for static devices in mainframe 0 SC device at ladd 0 in slod 0 SC device at ladd 8 in slot ? SC device in ladd 16 in slot 8 SC device at ladd 127 in slot 5 -- VXibus extender Searching for static devices on interconnect bus 127 SC device at ladd 128 in slot 0 -- VXibus extender Searching for static devices in mainframe 128 SC device at ladd 144 in slot 7 Searching for dynamic devices in mainframe 128 DC device in slot 3 moved to ladd 136, block size = 1 VXibus extender 128 Ladd window range: 128 to 159 INWARD VXibus extender 127 Ladd window range: 128 to 159, OUTWARD Searching for dynamic devices in mainframe 0 DC device in slot 3 moved to ladd 24, block size = 1</p>	<p>The resource manager identifies all statically configured modules, and then locates and configures all dynamically configurable modules.</p>
4	<p>Searching for pseudo devices</p>	<p>Pseudo devices are instruments such as IBASIC.</p>
5	<p>Configuring Commander/Servant hierarchy ladd = 0, cmdr ladd = -1 ladd = 8, cmdr ladd = 0 ladd = 16, cmdr ladd = 0 ladd = 24, cmdr ladd = 0 ladd = 136, cmdr ladd = 0 ladd = 144, cmdr ladd = 0 Validating Commander / Servant hierarchy Commander ladd 24 granted device ladd 32 Commander ladd 24 granted device ladd 64</p>	<p>The resource manager establishes the VXibus system's commander/servant hierarchies based on the commander's servant area and the servant's logical address.</p>
6	<p>Mapping A24 Memory Searching for A24 memory in mainframe 128 VXibus extender 128 A24 window range: 00000000 to 00FFFFFF, OUTWARD VXibus extender 127 A24 window range: 00000000 to 00FFFFFF, INWARD Searching for A24 memory in mainframe 0 ladd 0, offset = 00200000H, size = 131072 (bytes) Mapping A32 memory Searching for A32 memory in mainframe 128 VXibus extender 128 A32 window range: 00000000 to FFFFFFFF, OUTWARD VXibus extender 127 A32 window range: 00000000 to FFFFFFFF, INWARD Searching for A32 memory in mainframe 0</p>	<p>The resource manager allocates A24 addresses to access the memory located on the modules at logical addresses 0, 24, and 64. The offset is specified in hexadecimal and the size is specified in bytes. In this system, there are no A32 devices.</p>
7	<p>Configuring VME interrupts VME interrupt line 1 assigned to ladd 0, handler ID 1 VME interrupt line 2 assigned to ladd 24, handler ID 1 VME interrupt line 3 assigned to ladd 64, handler ID 1 VME interrupt line 4 - no handler assigned VME interrupt line 5 - no handler assigned VME interrupt line 6 - no handler assigned VME interrupt line 7 - no handler assigned VXibus extender 128 interrupts: 1-OUT 2-DIS 3-DIS 4-DIS 5-DIS 6-DIS 7-DIS VXibus extender 127 interrupts: 1-IN 2-DIS 3-DIS 4-DIS 5-DIS 6-DIS 7-DIS</p>	<p>The resource manager allocates interrupt lines to itself and to the other interrupt handlers in the system.</p>
8	<p>SYSTEM INSTALLED AT SECONDARY ADDR 0 VOLTMR INSTALLED AT SECONDARY ADDR 1 SWITCH INSTALLED AT SECONDARY ADDR 2 MBinstr INSTALLED AT SECONDARY ADDR 3 SYSTEM instrument started BNO issued to ladd 24, BNO response = FFFE Opening HP-IB access for message based device at sec addr 03</p>	<p>The resource manager identifies the secondary HP-IB addresses used in the system, starts the system instrument (i.e. Command Module), issues the Begin Normal Operation (BNO) command to its direct message based servant, and opens HP-IB access to the module at secondary HP-IB address 03.</p>

Figure 2-4. RM Configuration With Extenders

Reserving A24/A32 Address Space

As previously mentioned, the resource manager cannot determine when VME devices have been installed in the system. To prevent the resource manager from allocating A24/A32 addresses intended for VME devices to VXI devices, the address allocation table is used. The A24/A32 Address Allocation table is also used to assign different addresses to VXI devices than those (default) addresses assigned by the resource manager during power-on.

The A24/A32 Address Allocation Table

The A24/A32 Address Allocation table is created and stored in the Command Module as follows:

1. Table space in the Command Module's non-volatile user RAM is made available by allocating a segment of RAM with the command:

DIAGnostic:NRAM:CREate < size >

2. The location (starting address) of the table in RAM is determined with the command:

DIAGnostic:NRAM:ADDress?

3. Data is downloaded into the table with the command:

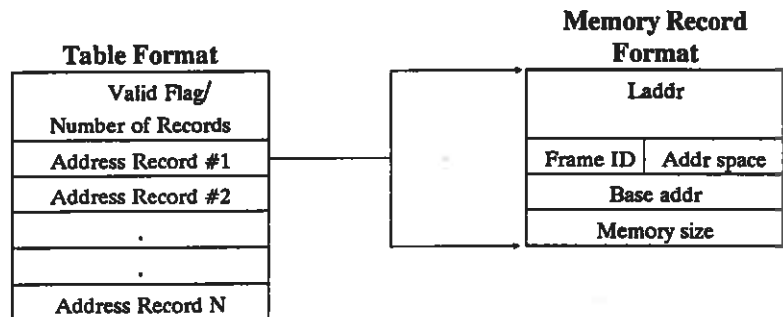
DIAGnostic:DOWNload < address > < data >

4. The table is linked to the appropriate algorithm in the Command Module processor with the command:

VXI:CONFigure:MTABLE < address >

Table Format The format of the A24/A32 Address Allocation table is shown in Table 2-3.

Table 2-5. A24/A32 Address Allocation Table Format.



The table parameters are:

- **Valid Flag (0|1):** 1 indicates the table is valid and the addresses reserved accordingly. 0 will cause an error message (Error 43). Valid Flag is part of the table header and is represented by the upper eight bits of the header word.
- **Number of Records:** is the number of address records in the table. You must have one record for each VME or VXI device for which memory is reserved. Number of Records is part of the table header and is represented by the lower eight bits of the header word.
- **Laddr:** is the logical address of the VXI device for which A24/A32 addresses are reserved. -1 specifies a VME device. Field is one word.
- **Addr space(24|32):** is the address space being reserved. 24 specifies A24 addresses are being reserved. 32 specifies A32 addresses are being reserved. Field is one word.
- **Frame ID (0-255):** is the logical address of the slot 0 commander for the mainframe containing the VME memory block (8 bit byte). This field must be included but is ignored if the logical address field is not -1 (indicating a VME device).
- **Base addr (0 to $2^{24} - 1$ / 0 to $2^{32} - 1$):** is the starting address (offset) of the A24 or A32 addresses to be reserved. Field is two words (4 bytes) and is specified in decimal.
- **Memory size (1 to $2^{24} - 1$ / 1 to $2^{32} - 1$):** is the amount of memory for which addresses must be reserved. This field must be specified but is ignored if a VXI A24/A32 device is specified (Laddr). Field is two words (4 bytes) and is specified in decimal.

Determining the Table Size

The A24/A32 address allocation table has a one word header, the first two entries in the address record are one word each, and the second two entries are two word each. The amount of RAM allocated with DIAG:NRAM:CRE is specified in bytes. Since one word is two bytes, the amount of RAM to allocate is computed as:

$$2 + 12(N)$$

where 2 is the two byte header, 12 is the number of bytes per address record (2 + 2 + 4 + 4), and N is the number of address records. For example, to reserve A24 addresses for two VME devices, the table size would be: $2 + 12(2) = 26$ bytes. DIAG:NRAM:CRE would be executed as:

```
OUTPUT @E1405;"DIAG:NRAM:CRE 26"
```

Data Format

Data is sent to the A24/A32 address allocation table in 16-bit words. This is accomplished by reading the data into an Integer variable in the computer, and then downloading the data to the table using the ANSI/IEEE 488.2-1987 Arbitrary Block Program Data format. This process is shown in the example program which follows. More information on the Arbitrary Block Program format can be found in Chapter 5, and in the ANSI/IEEE 488.2-1987 document.

The Table Header The table header is sent as a single 16-bit word which must contain the Valid Flag and the number of address records. For a valid table, the header is 256 plus the number of records. For example, to indicate a valid table with two records, the header is 258 (256 + 2).

CAUTION

When downloading data into the A24/A32 address allocation table, DIAGnostic:DOWNload does not determine if the table is large enough to store the data. If the amount of data sent by DIAGnostic:DOWNload is greater than the (table) space allocated by DIAGnostic:NRAM:CREate, system errors will occur. You can recover from these errors by executing DIAG:BOOT:COLD or by pressing "Ctrl R" on an RS-232 terminal while cycling mainframe power.

**Example: Reserving
A24 Addresses for a
VME Device**

The following program reserves a block of A24 addresses for a VME device. The program assumes the device has been configured with a starting A24 address of 300000₁₆ and a size of 80000₁₆.

Again, this procedure is used when you want to reserve a specific block of A24/A32 addresses for a VME device, or when you want to assign addresses to a VXI device that are different from those assigned by the resource manager.

NOTE

The computer syntax shown is for an external controller or for an embedded (HP V/360) controller.

```
10 !Assign I/O path and allocate variable to store A24/A32 memory
20 !allocation data to be downloaded to the command module.
30 ASSIGN @E1405 TO 70900;EOL CHR$(10) END
40 INTEGER Mem_alloc(1:7)
50 !
60 !Allocate a segment of non-volatile user RAM on the Command
70 !Module to store the A24/A32 memory allocation table.
80 OUTPUT @E1405;"DIAG:NRAM:CRE 14"
90 !
100 !Re-start the system instrument to allocate the user RAM. Wait for the
110 !re-start to complete before continuing.
120 OUTPUT @E1405;"DIAG:BOOT:WARM"
130 ON TIMEOUT 7,.1 GOTO Complete
140 Complete: B = SPOLL(70900)
150 OFF TIMEOUT 7
```



```

160 !
170 !Return the starting address of the table in non-volatile user RAM.
180 OUTPUT @E1405;"DIAG:NRAM:ADDR?"
190 ENTER @E1405;A
200 !
210 !Download the following: the table is valid, there is one memory
220 !record: logical address is -1 (VME card), A24 address space (24)
230 !base address is 300000h (48,0), and memory size is 80000h (8,0).
240 !See Comments.
250 DATA 257,-1,24,48,0,8,0
260 READ Mem_alloc(*)
270 OUTPUT @E1405 USING "#,3(K)";"DIAG:DOWN ";A;" ,#0"
280 OUTPUT @E1405 USING "W";Mem_alloc(*)
290 !
300 !Link the A24/A32 memory allocation table to the appropriate
310 !algorithm.
320 OUTPUT @E1405;"VXI:CONF:MTAB ";A
330 !
340 !Re-start the system instrument to set the user-defined configuration.
350 OUTPUT @E1405;"DIAG:BOOT:WARM"
360 END

```

Comments

- To download the base address and memory size (line 270) they must each be specified as two 16-bit words (line 250). This can be accomplished as follows:

Memory Size: 300000 ₁₆ =	0030	0000
	1st word	2nd word
	48 ₁₀	0 ₁₀
Memory Size: 80000 ₁₆ =	0008	0000
	1st word	2nd word
	8 ₁₀	0 ₁₀

- The following errors are associated with A24/A32 address allocation table:

ERROR 8: INACCESSIBLE A24 MEMORY

This error occurs when all or part of an A24 device overlaps the top 2 MBytes or bottom 2 MBytes of the A24 address space. This space becomes inaccessible to the Command Module.

ERROR 32: INACCESSIBLE A32 MEMORY

This error occurs when all or part of an A32 device overlaps the top 500 MBytes or bottom 500 MBytes of the A32 address space.

ERROR 33: INVALID UDEF MEMORY BLOCK

This error occurs when an invalid base address is specified, or when the size of the memory exceeds the A24 or A32 address space (given the base address specified).

ERROR 34: UDEF MEMORY BLOCK UNAVAILABLE

This error occurs when the same base address and memory size are specified more than once in the same table, or if addresses in the specified block are already used.

ERROR 35: INVALID UDEF ADDRESS SPACE

This error occurs when the address space (Addr space) specified in the table is A24 and an A32 device is installed, or vice versa.

ERROR 36: DUPLICATE UDEF MEMORY LADD

This error occurs when a logical address is specified more than once in the same table. This does not apply to VME devices (address = -1).

ERROR 43: INVALID UDEF MEM TABLE

This error occurs when the table's valid flag is set to 0.

ERROR 44: INVALID UDEF MEM TABLE DATA

This error occurs when an invalid logical address is specified.

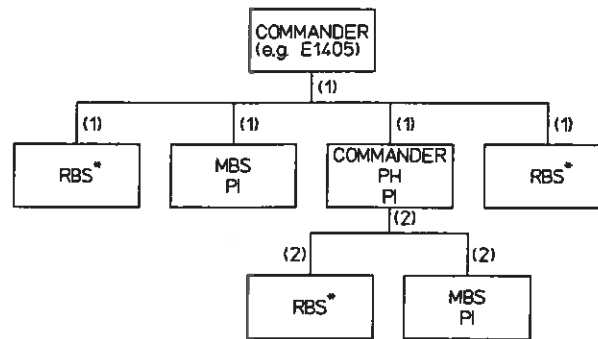
- The A24/A32 addresses reserved by the A24/A32 address allocation table are reserved within the system until DIAGnostic:BOOT:COLD is executed, or until VXI:CONFigure:MTABLE 0 is executed.

Interrupt Line Allocation

In a VXIbus system, communication and coordination between a commander module and its servant module(s) is often achieved using the VXIbus backplane interrupt lines. During the configuration sequence, the resource manager assigns interrupt lines to programmable interrupt handler modules and interrupter modules.

Both commanders and servants can be interrupt handlers and/or interrupters. The Command Module which is a programmable interrupt handler, is not an interrupter. Thus, in systems where the Command Module is a servant to another commander, it communicates with the commander through its Response and Data Low registers (see the VXIbus System Specification).

The assignment and use of the interrupt lines is described in Figure 2-4 and with the information which follows.



* MODULES WHICH ARE NOT PROGRAMMABLE INTERRUPTERS MUST SELECT THE INTERRUPT LINE USING THE JUMPERS ON THE MODULE.

MBS - MESSAGE BASED SERVANT

RBS - REGISTER BASED SERVANT

PI - PROGRAMMABLE INTERRUPTER

PH - PROGRAMMABLE INTERRUPT HANDLER

(1) - INTERRUPT LINE 1

(2) - INTERRUPT LINE 2

Figure 2-5. Example Interrupt Line Allocation

Note the following regarding interrupt line allocation:

- There are seven VXIbus backplane interrupt lines. As the resource manager, the HP E1405 Command Module assigns itself interrupt line 1 (default). Additional interrupt lines (up to all seven) can be assigned to the Command Module using the Interrupt Line Allocation Table. Interrupt lines not assigned to programmable handlers remain unassigned.
- Many Hewlett-Packard modules have interrupt line 1 as their factory setting. Thus, they are available for immediate use with the HP E1405 Command Module.
- Commander modules which are programmable interrupt handlers are assigned interrupt lines 2, 3, 4,...7; beginning with the commander with the lowest logical address. Only one interrupt line is assigned per interrupt handler.
- Servant modules which are programmable interrupt handlers are also assigned interrupt lines, beginning with the servant with the lowest logical address. Only one interrupt line is assigned per interrupt handler.
- Servant modules which are programmable interrupters are assigned the same interrupt line assigned to their commander.
- For modules which are not programmable, the interrupt line is selected using jumpers on the modules. The Interrupt Line Allocation Table is used to tell the Command Module which line was selected.

User-Defined Interrupt Line Allocation

The interrupt line allocation table allows you to assign additional interrupt lines to a specific handler, reserve interrupt lines for non-programmable interrupt handlers and interrupters, and assign lines to VME devices.

The Interrupt Line Allocation Table

User-defined interrupt line allocations are specified with an interrupt line table created in the Command Module. The table is created as follows:

1. Table space in the Command Module's non-volatile user RAM is made available by allocating a segment of RAM with the command:

DIAGnostic:NRAM:CREate < size >

2. The location (starting address) of the table in RAM is determined with the command:

DIAGnostic:NRAM:ADDress?

3. Data is downloaded into the table with the command:

DIAGnostic:DOWNload < address > < data >

4. The table is linked to the appropriate algorithm in the Command Module processor with the command:

VXI:CONFigure:ITABLE < address >

Table Format The format of the interrupt line table is shown in Table 2-4.

Table 2-6. Interrupt Line Allocation Table Format

Table Format		Data Record Format
Valid Flag/ Number of Records	—	Intr Line
Data Record #1		Handler Laddr
Data Record #2		Number of Interrupters
.		Intr #1 Laddr
.		Intr #2 Laddr
Data Record #7		Intr M Laddr

The table parameters are:

- **Valid Flag (1/0):** 1 indicates the table is valid and the modules should be configured accordingly. 0 will cause an error message (Error 41). Valid Flag is part of the table header and is represented by the upper eight bits of the header word.
- **Number of Records (1 - 7):** is the number of data records in the table. A data record is required for each interrupt line assigned. Number of Records is part of the table header and is represented by the lower eight bits of the header word.
- **Intr Line (1 - 7):** is the interrupt line to be assigned to the programmable interrupt handler or interrupter, or the line reserved for a non programmable interrupter/handler or VME device. Field is one word.
- **Handler Laddr:** is the logical address of the programmable handler which will handle interrupts on the line specified by **Intr Line**. If -1 is specified, the line is reserved and no handler is assigned. The field is one word.
- **Number of Interrupters:** is the number of programmable interrupters on the interrupt line specified by **Intr Line**. If 0 is specified, there are no programmable interrupters. This reserves the line for a non-programmable interrupter. The field is one word.
- **Intr Laddr:** is the logical address of the programmable interrupter on the interrupt line specified. The logical address of each programmable interrupter on the line must be specified. Programmable interrupters can be assigned to interrupt lines with no handler. This allows a programmable interrupter to have a non-programmable interrupt handler handle its interrupts. If **Number of Interrupters** is 0, **Intr Laddr** is not specified.

Determining the Table Size

The interrupt line allocation table has a one word header and each data record contains three words, plus one word for each programmable interrupter logical address specified. The amount of RAM allocated with **DIAG:NRAM:CRE** is specified in bytes. Since one word is two bytes, the amount of RAM to allocate is computed as:

$$2 + 6(N) + 2 \sum_0^N M$$

where 2 is the two byte header, 6 is the number of bytes/data record, N is the number of data records (i.e. interrupt lines) and M is the number of programmable interrupters per data record. For example, to create a table for the following:

one interrupt handler

two interrupt lines

one interrupter on one line, three interrupters on second line

the table size would be:

$$\begin{array}{r} 2 + 6(2) + 2(4) = 22 \text{ bytes} \\ \quad | \quad \quad | \\ \quad (2 \text{ records}) \quad (4 \text{ interrupters}) \end{array}$$

DIAG:NRAM:CRE would be executed as:

OUTPUT @E1405;"DIAG:NRAM:CRE 22"

NOTE

When assigning an additional interrupt line to an interrupt handler, you must specify each line. Otherwise, the table will overwrite the line currently assigned, giving the handler only one line. For example, if the resource manager assigns interrupt line 2 to a handler and you want to also assign line 3 to the handler, lines 2 and 3 must be specified in the table. See "Example: Assigning an Interrupt Line".

Data Format

Data must be sent to the interrupt line allocation table in 16-bit words. This is accomplished by reading the data into an Integer variable in the computer, and then downloading the data to the table using the ANSI/IEEE 488.2-1987 Arbitrary Block Program Data format. This process is shown in the example program which follows. More information on the Arbitrary Block Program format can be found in Chapter 5, and in the ANSI/IEEE 488.2-1987 document.

The table header is sent as a single 16-bit word which must contain the Valid Flag and the number of data records. For a valid table, the header is 256 plus the number of data records. For example, to indicate a valid table with one data record, the header is 257 (256 + 1 = 257).

CAUTION

When downloading data into the interrupt line allocation table, DIAGnostic:DOWNload does not determine if the table is large enough to store the data. If the amount of data sent by DIAGnostic:DOWNload is greater than the (table) space allocated by DIAGnostic:NRAM:CREate, system errors will occur. You can recover from these errors by executing DIAG:BOOT:COLD, or by pressing "Ctrl R" on an RS-232 terminal while cycling mainframe power.

Example: Assigning an Interrupt Line

The following example shows how an additional interrupt line is assigned to a programmable interrupt handler and reserved for a non-programmable interrupter (Figure 2-5).

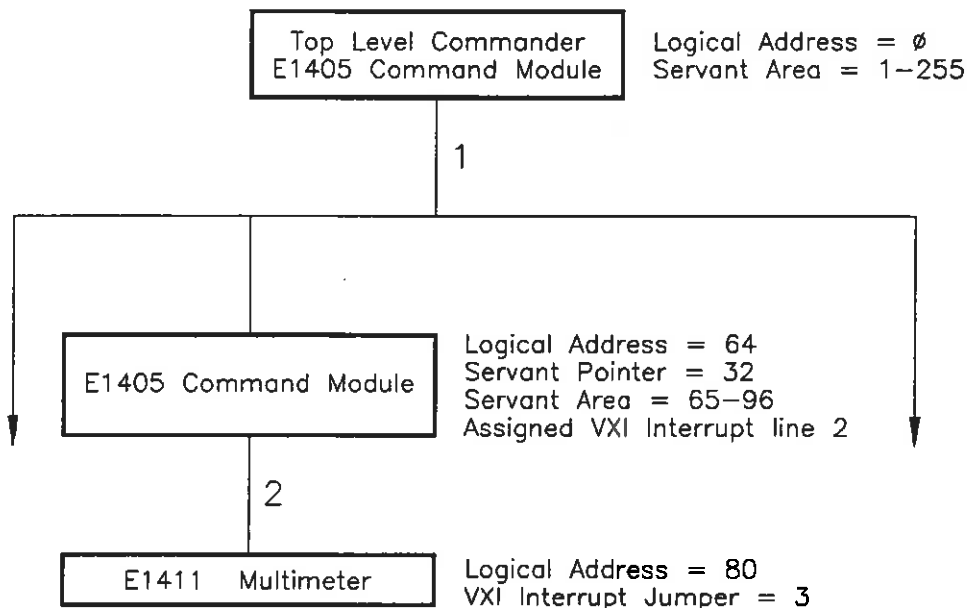


Figure 2-6. Assigning an Additional Interrupt Line

D:\DWGS\F_2_5

The program assumes that a VXibus system contains an HP E1411 5 1/2-Digit Multimeter that is a servant to a second E1405 Command Module at logical address 64. Since the E1405 is the only other commander and is a programmable interrupt handler, it is assigned interrupt line 2 by the resource manager. The E1411, however, has its interrupt jumper set for line 3. For the E1411 to communicate with the Command Module, the Command Module must also be assigned to handle interrupt line 3.

NOTE

The computer syntax shown is for an external controller or for an embedded (HP V/360) controller.

```

10 !Assign an I/O path and allocate a variable to store interrupt
20 !line data to be downloaded to the Command Module.
30 ASSIGN @E1405 TO 70900;EOL CHR$(10) END
40 INTEGER Intr_line(1:7)
50 !
60 !Allocate a segment of non-volatile user RAM on the Command
70 !Module to store the interrupt line table (2 data records, no
interrupters).
80 OUTPUT @E1405;"DIAG:NRAM:CRE 14"
90 !

```

```

100 !Re-start the system instrument to define the user RAM. Wait for the
110 !re-start to complete before continuing.
120 OUTPUT @E1405;"DIAG:BOOT"
130 ON TIMEOUT 7,.1 GOTO Complete
140 Complete: B = SPOLL(70900)
150 OFF TIMEOUT 7
160 !
170 !Return the starting address of the non-volatile user RAM.
180 OUTPUT @E1405;"DIAG:NRAM:ADDR?"
190 ENTER @E1405;A
200 !
210 !Download the following: the table is valid - there are two data
records.
220 ! Interrupt line 3 (and line 2) is assigned to the handler at logical
230 !address 64. There are no programmable interrupters on either line.
240 DATA 258,2,64,0
250 DATA 3,64,0
260 READ Intr_line(*)
270 OUTPUT @E1405 USING "#,3(K)";"DIAG:DOWN ";A;" ,#0"
280 OUTPUT @E1405 USING "W";Intr_line(*)
290 !
300 !Link the interrupt line table to the appropriate algorithm.
310 OUTPUT @E1405;"VXI:CONF:ITAB ";A
320 !
330 !Re-start the system instrument to set the user-defined configuration.
340 OUTPUT @E1405;"DIAG:BOOT"
350 END

```

Comments

- Although interrupt line 2 was assigned to the Command Module at logical address 64 by the resource manager, the line must be "re-assigned" when line 3 is assigned. Otherwise, line 3 will be assigned in place of line 2.
- The interrupt lines assigned by the interrupt line table are used by the system until DIAGnostic:BOOT:COLD is executed.
- When using multiple Command Modules, HP-IB cables must be connected from the slot 0 Command Module, to each Command Module in the system.
- In this program, the Command Module at logical address 64 has a primary HP-IB address of 08. It has a servant pointer setting of 32, thus its servant area is from logical address 65 to logical address 96. If the E1411 multimeter has a logical address of 80, its secondary HP-IB address is 10. Thus, when programming this multimeter, its HP-IB address is:

```
OUTPUT 70810;"....
```

When programming this Command Module, its HP-IB address is:

```
OUTPUT 70800;"...
```


- The following errors are associated with interrupt line allocation table:

ERROR 24: INTERRUPT LINE UNAVAILABLE

This error occurs when an interrupt line is already assigned.

ERROR 25: INVALID UDEF HANDLER

This error occurs when the logical address specified for the interrupt handler (Handler Laddr) is a device that is not an interrupt handler.

ERROR 26: INVALID UDEF INTERRUPTER

This error occurs when the logical address specified for the interrupter (Intr # Laddr) is a device that is not an interrupter.

ERROR 41: INVALID UDEF INTR TABLE

This error occurs when the table's valid flag does not equal 1.

ERROR 42: INVALID UDEF INTR TABLE DATA

This error occurs when: there is an invalid number of records (valid numbers are 1 to 7), there is an invalid interrupt line number (valid numbers are 1 to 7), there is an invalid interrupt handler logical address (valid addresses are 0 to 255), or there is an invalid interrupter logical address (valid addresses are 0 to 255).

- The interrupts assigned by the cInterrupt table are used by the system until DIAGnostic:BOOT:COLD is executed, or until VXI:CONFigure:ITABLE 0 is executed.

Starting System Operation

The resource manager completes the configuration sequence by issuing the "Begin Normal Operation" (BNO) command to all top level commanders and to each of its direct message based servants. BNO is not sent to register based modules. The module receiving BNO responds by writing its status to the Data Low register which is read by the resource manager. More information on BNO and on the Data Low register can be found in the VXIbus System Specification.

If the Command Module is in a system where it is not the resource manager, it sends BNO to each of its message based servants once it receives BNO from its commander.

Using the Display Terminal Interface

About this Chapter

This chapter shows you how to use the HP E1405 Command Module's Display Terminal Interface to operate instruments in a Series C mainframe. The terminal interface uses the built-in RS-232 port and/or the optional HP E1324A Datacomm Module to provide a "front panel" for C-size VXIbus systems.

The main sections of this chapter include:

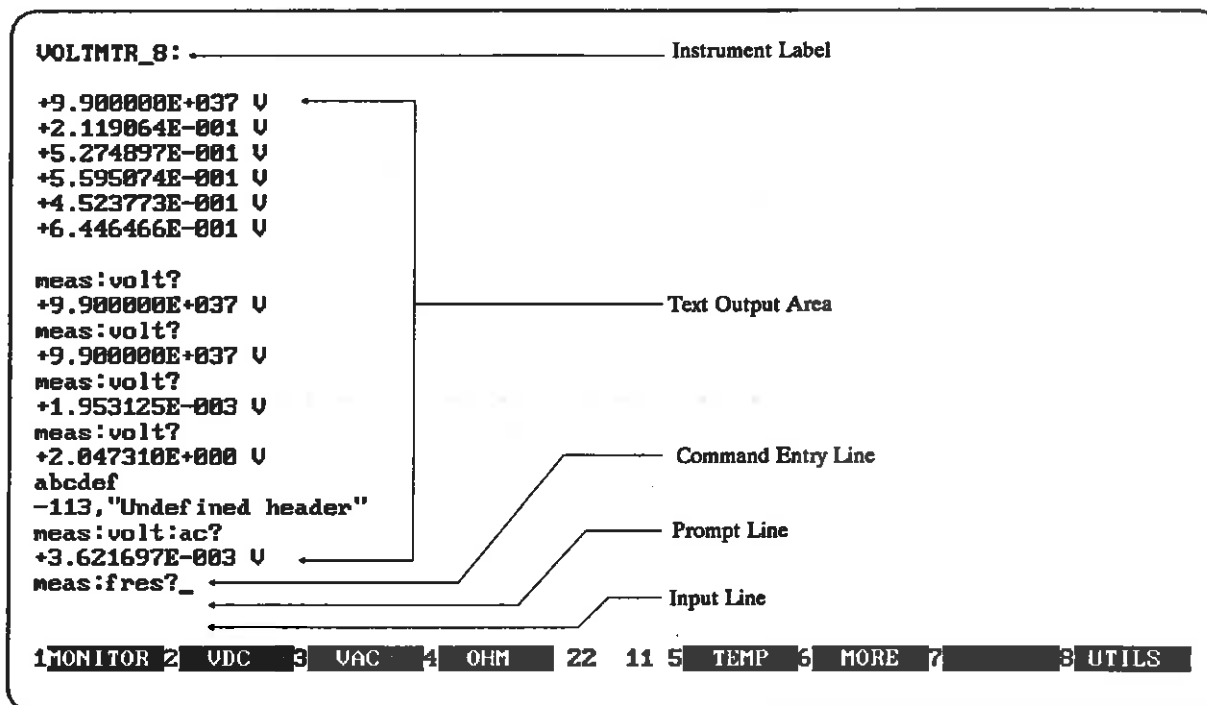
- Terminal Interface Features 3-1
- Using Menus 3-2
- Executing Commands 3-17
- General Key Descriptions 3-18
- Using Supported Terminals 3-20
- Using Other Terminals 3-24
- In Case of Difficulty 3-27
- Instrument Menus 3-29

NOTE

This chapter discusses *using* the display terminal interface. It assumes that you have already connected your terminal and configured it to communicate with the Command Module. For information on connecting and configuring your terminal, refer to the C-Size VXIbus Systems "Installation and Getting Started Guide".

Terminal Interface Features

Figure 3-1 shows a typical terminal interface display with its function labels across the bottom of the screen. The first five function keys (f1 through f5) select instrument menu choices. Function keys f6 through f8 provide menu control and access to utility functions. The tutorials in this chapter show how to use most of the menu control and utility function keys. See "General Key Descriptions" for a complete description of each of these key functions.



- Notes:**
1. Example screens are from HP AdvanceLink terminal emulator.
 2. Later screen examples are shown compressed (only 4 lines tall) and may show only part of the screen width.

Figure 3-1. Typical Terminal Interface Display

Using Menus

A System Instrument menu and a variety of other instrument menus (depending on the instruments in the Command Module servant area) are available from the terminal interface. These menus incorporate the most used functions but do not provide access to the complete functionality of an instrument. If a particular function is not available from a menu, you can type the corresponding Common Command or SCPI command string and execute it from the terminal interface. See "Executing Commands" later in this chapter for more information.

When you select an instrument, you are assigning the terminal interface to that instrument. This means that any menu operations, commands executed or recalled, errors displayed, etc. pertain only to that instrument. Terminal interface operation of an instrument is independent from other instruments and independent from the remote operation of the instrument. To operate another instrument from the terminal interface, you must select that instrument.

How Instruments Appear In the Menu

Instruments in the terminal interface menu are register-based devices which are in the servant area of the Command Module. Message-based devices, or register-based devices outside the Command Module's servant area, do not appear in the menu.

NOTE

Message-based instruments, which do not appear in instrument menus, can be programmed using the SYSTEM instrument menu. See "Using the System Instrument Menu" later in this chapter.

Multiple Command Modules

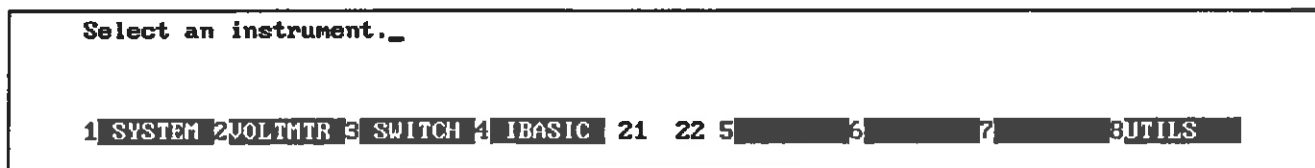
In systems with multiple Command Modules, the instruments in the menu depend on the Command Module whose RS-232 port is connected to the terminal. To change menus (Command Modules):

1. Move the RS-232 cable to the desired Command Module.
2. Type CTRL-D to guarantee that the display terminal interface is in control of the terminal.
3. Type:
 - ST (followed by Return) for auto-identification of the terminal.
 - or
 - ST HP (followed by Return for HP terminals - 700/94, 700/92, 26xx, 23xx)
 - or
 - ST HP70043 (followed by Return for the HP 700/43 terminal)
 - or
 - ST VT100 (followed by Return for VT100[®] emulators)
 - or
 - ST VT220 (followed by Return for VT220[®] emulators)
 - or
 - ST WYSE30 (followed by Return for WY-30[®] emulators)
 - or
 - ST WYSE50 (followed by Return for WY-50[®] emulators)

This changes the menu to correspond to the instruments in the servant area of the new Command Module.

A 60-Second Menu Tutorial

Following the power-on sequence or a system reset, the screen shows the *Select an instrument* menu (see Figure 3-2). This menu allows you to select one of the instruments listed.



Note: Typical instruments shown. Actual choices depend on installed instruments.

Figure 3-2. "Select an instrument" Menu

The menu select and menu control function keys (usually labeled **f1 - f8** on their key caps) are defined by eight function labels located across the bottom of the terminal screen. Once you learn how these keys operate, using the menus is easy (key labels are shown in bold text in this chapter):

To select a displayed menu choice, press the function key (**f1 - f5**) which corresponds to the function key label.

- When there are more than five menu choices, function key **f6** becomes labeled **MORE**. Press **MORE** to display the next group of choices. By repeatedly pressing **MORE** you can display all groups of choices. After you have displayed all groups of choices, pressing **MORE** again returns to the first group of choices.
- Whenever the screen is requesting information (input prompt) such as *Enter the device's logical address*, just type the information and press **Return** (may be **Enter** on a terminal emulator).

If you pressed the wrong menu key and do not want to enter the requested information, you can escape the input prompt and stay at the same menu level by pressing **ESC** or **PRV_MENU**.

If you make an incorrect entry in response to an input prompt, the bottom line of the Text Output Area will show an error message. When this happens, just select that menu choice again (**f1 - f5** keys), re-type the correct information, and press **Return**.

- Press **PRV_MENU** or **ESC** to return to the previous menu within an instrument menu or escape from an input prompt. Press **SEL_INST** to return to the *Select an Instrument* menu (see next item). Note that when you leave an instrument and return later, you return to the same menu location you were at when you left. Any information below the Text Output Area will also be re-displayed when you return.
- In addition to the instrument menu keys, **CLR_INST**, **RST_INST** and **SEL_INST** are helpful when operating instruments. These and other utility keys are accessed by pressing the **UTILS** key. See "Executing Commands" for information on the **RCL_....** keys in this menu.
 - **CLR_INST** clears the instrument's terminal interface input and output buffers (remote buffers are not cleared) and returns to the top level of the instrument menu. Press **CLR_INST** whenever an instrument is busy, is not responding to terminal interface control, or to abort a command being entered from the terminal interface.
 - **RST_INST** clears all terminal interface and remote input and output buffers and resets the instrument.
 - **SEL_INST** returns you to the *Select an Instrument* menu. **SEL_INST** is the key "under" the **UTILS** key. You can easily return to the *Select an Instrument* menu by pressing **f8** twice.

How to Access the Utility Keys

VOLTMTR_B:

-

1 MONITOR 2 VDC 3 VAC 4 OHM 22 1 5 TEMP 6 MORE 7 8 UTILS

VOLTMTR_B:

-

1 RST INST 2 CLR INST 3 4 22 1 5 RCL MENU 6 RCL PREV 7 RCL NEXT 8 SEL INST

Using the System Instrument Menu

The System Instrument menu allows you to:

- Display logical address and instrument information
- Read the Command Module HP-IB address
- Configure the RS-232 port
- Program message-based devices
- Set the system clock and calendar
- Reset the Command Module

The menus on the following pages demonstrate how to do each of the above.

How to Display Logical Addresses and Instrument Information

```
Select an instrument._
1 SYSTEM 2 VOLTMR 3 SWITCH 4 IBASIC
```

```
SYSTEM_0:
-
1 CONFIG? 2 HP-IB? 3 RS-232 4 DEBUG
```

```
SYSTEM_0:
-
1 LADDRS 2 DEVICE 3 _____ 4 _____
```

Enter device's logical address and press Return for Individual instrument information, or just enter one space and Return, for information on all instruments.
(In this case, 8 was entered)

```
SYSTEM_0:
+0,+0,+32,+240
1 LADDRS 2 DEVICE 3 _____ 4 _____
```

Logical address of selected device

Instrument name

```
SYSTEM_0:
+0,+0,+4095,+65344,-1,+0,REG,A16 ,#00000000,#00000000,READY,"", "", "", "VOLTMR
INSTALLED AT SECONDARY ADDR 1"
1 LADDRS 2 DEVICE 3 _____ 4 _____ 24 1 5 _____ 6 _____ 7 PRU_MENU8 UTILS
```

HP-IB secondary address

Note: For a description of each field of the instrument information, see VXI:CONF:DLIS? in the SCPI Command Reference section.

How to Read the Command Module HP-IB Address

Select an instrument. _

1 SYSTEM 2 VOLTMR 3 SWITCH 4 IBASIC

SYSTEM_0:

-

1 CONFIG? 2 HP-IB? 3 RS-232 4 DEBUG

Typical HP-IB address: +9
SCPI command used:
SYST:COMM:GPIB:ADDR?

Configuring the Command Module's RS-232 Port

Select an instrument._

1 SYSTEM 2 VOLTMR 3 SWITCH 4 IBASIC

SYSTEM_0:

1 CONFIG? 2 HP-IB? 3 RS-232 4 DEBUG

SYSTEM_0:

1 BAUD 2 PARITY 3 BITS 4 PACE 53 1 5 CONTROL 6 MORE 7 PRU_MENU 8 UTILS

Note: Configuration of the Command Module's RS-232 port is covered in the C-Size VXibus Systems "Installation and Getting Started Guide".

Programming Message-Based Devices

Select an instrument._
1 SYSTEM 2 VOLTMR 3 SWITCH 4 IBASIC

SYSTEM_0:
-
1 CONFIG? 2 HP-IB? 3 RS-232 4 DEBUG

SYSTEM_0:
-
1 READ 2 WRITE 3 SEND 4 RECEIVE 53 1 5 RESET 6 MORE 7 PRV MENU 8 UTILS

SYSTEM_0:
Enter logical address [,termination option]

SCPI command used
VXI:REC? <laddr>

SYSTEM_0:
-
1 MESSAGE 2 COMMAND 3 QUERY 4

SYSTEM_0:
Enter logical address, string [,termination option]
1 2 3 4 54 1 5 6

SCPI command used:
VXI:SEND <laddr>,"<string>"

Setting the System Clock and Calendar

Select an instrument. _

1 SYSTEM 2 VOLTMR 3 SWITCH 4 IBASIC

SYSTEM_0:

-
1 CONFIG? 2 HP-1B? 3 RS-232 4 DEBUG 53 1 5 TIME 6 MORE

SYSTEM_0:

-
1 READ 2 SET 3 4

SYSTEM_0:

-
1 DATE 2 RESET 3 4

SYSTEM_0:

-
1 READ 2 SET 3 4

SYSTEM_0:

Enter date (yyyy,mm,dd)

1 2 3 4

SCPI command used

SYST:DATE <date>

SYSTEM_0:

Enter time (hh,mm,ss)

1 2 3 4

SCPI command used:

SYST:TIME <time>

How to Reset the System

Select an instrument._

1 SYSTEM 2 VOLTMR 3 SWITCH 4 IBASIC 21 22 5 6 7 8 UTILS

SYSTEM_0:

-

1 CONFIG? 2 HP-IB? 3 RS-232 4 DEBUG 53 1 5 TIME 3 MORE 7 8 UTILS

SYSTEM_0:

-

Press f2 to Reset

1 DATE 2 RESET 3 4 53 1 5 6 MORE 7 8 UTILS

Using the Other Instrument Menus

The instrument menus allow you to access the most-used instrument functions or to monitor an instrument (monitor mode) while it is being controlled from remote. The Switchbox menu is used to show you how to use the instrument menus. Menus are available for many but not all instruments. See “Instrument Menus”, later in this chapter, for more information on a particular instrument’s menu. The Switchbox menu allows you to:

- Open and close channels
- Scan channels
- Show Module data (type and description)
- Monitor a switchbox
- Reset a selected switch module

Selecting the Switchbox

To select the Switchbox, press the function key (f1 - f5) which corresponds to the label SWITCH in the “*Select an instrument*” menu. (If the “*Select an instrument*” menu is not being displayed press UTILS then SEL_INST.)

NOTE

After you press the function key for SWITCH, the screen may show: “*Select SWITCH at logical address: _*” while the function key labels show two or more logical addresses. This means more than one Switchbox is installed in the mainframe. To select one of the Switchboxes, press the function key for the logical address key label.

The charts on the following pages show how to use the Switchbox menu. Keep the following points in mind when using the menu:

- The card number identifies a module within the Switchbox. The module with the lowest logical address is always card number 01. The module with the next successive logical address is card number 02 and so on.
- The @ character is required preceding a channel list when executing a Switchbox command from the terminal interface or remote. When entering a channel list in response to a menu prompt however, do not precede it with the @ character. Doing so causes a syntax error.

How to Open/Close Channels

Switchbox instrument at logical address 32
(secondary address = 04)

```
SWITCH_32:  
-  
1 MONITOR 2 OPEN 3 CLOSE 4 SCAN
```

```
SWITCH_32:  
Enter channel list  
1 2 3 4
```

SCPI command used:
OPEN <channel_list>

Enter Channel List and press Return
(e.g., 102 for channel 2 on card #1)

```
SWITCH_32:  
Enter channel list  
1 2 3 4
```

SCPI command used:
CLOSE <channel_list>

How to Scan Channels

```
SWITCH_32:  
-  
1 MONITOR 2 OPEN 3 CLOSE 4 SCAN
```

```
SWITCH_32:  
-  
1 SET UP 2 STEP 3 4
```

Press f2 to advance to the next channel in
the Scan List (i.e. to trigger the instrument.)

```
SWITCH_32:  
Enter channel list  
1 2 3 4
```

Enter Channel List and press Return
(e.g., 100:115 to scan channels 00 to 15 on card #1)

How to Display Card Type , Description, or Reset Card

```
SWITCH_32:
-
1 MONITOR 2 OPEN 3 CLOSE 4 SCAN 23 1 5 CARD 3
```

```
SWITCH_32:
-
1 TYPE? 2 DESCR? 3 RESET 4
```

```
SWITCH_32:
Enter card number
1 2 3 4
```

Enter Card Number and press Return

```
SWITCH_32:
HEWLETT-PACKARD ,E1345A,0,A.03.00
1 TYPE? 2 DESCR? 3 RESET 4
```

SCPI command used:
SYST:CTYP <card_number>

```
SWITCH_32:
Enter card number
1 2 3 4
```

Enter Card Number and press Return

SCPI command used:
SYST:CPON <card_number>

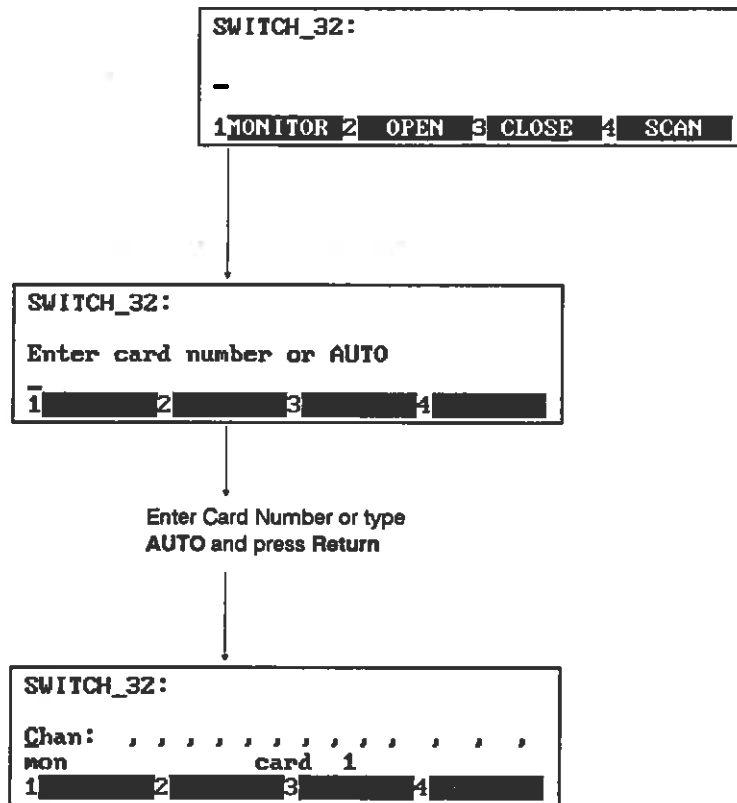
```
SWITCH_32:
Enter card number
1 2 3 4
```

Enter Card Number and press Return

```
SWITCH_32:
"16 Channel Relay Mux"
1 TYPE? 2 DESCR? 3 RESET 4
```

SCPI command used:
SYST:CDES <card_number>

How to Select Monitor Mode



SCPI commands used:
DISP:MON:CARD <card_number>
DISP:MON:STAT ON

Monitor Mode Monitor mode displays the status of an instrument while it is being controlled from remote. Monitor mode is useful for debugging programs. You can place an instrument in monitor mode using terminal interface menus, or by executing the **DISP:MON:STAT ON** command from the terminal interface or by remote. Pressing most terminal interface keys will automatically exit monitor mode and return to the instrument menu. However, you can use the left and right arrow keys in monitor mode to view long displays.

NOTE

Enabling monitor mode slows instrument operations. If the timing or speed of instrument operations is critical (such as making multimeter readings at a precise time interval), you should not use monitor mode.

Table 3-1 shows the status annunciators that may appear in the bottom line of the screen in monitor mode. Some instruments also have device-specific annunciators (see the plug-in module manual for more information).

Table 3-1. Monitor Mode Display Annunciators

Annunciator	Description
mon	The instrument is in monitor mode
busy	The instrument is executing a command
err	An error has occurred (see "Reading Error Messages" below)
srq	A service request has occurred

Reading Error Messages

Whenever the screen is showing the *err* annunciator, an error has occurred for the instrument being monitored. You can read the error message, although doing so cancels monitor mode. To read an error message, type the following SCPI command (followed by the Return key):

SYST:ERR?

The error message will be displayed in the bottom line of the Text Output Area. To see if another error was logged, repeat the above command by pressing **UTILS**, **RCL_PREV**, then **Return**.

After you have read all the error messages, executing the **SYST:ERR?** command causes the screen to show: **+0,"No error"**. After reading the error message(s), press **f1** to return to monitor mode.

Executing Commands

From the terminal interface, you can type and execute IEEE 488.2 Common Commands and SCPI Commands for the instrument presently selected by the *Select an instrument* menu. (However, you cannot execute a command when the screen is requesting that you input information.) This is particularly useful for accessing functions not available in an instrument's menu. For example, assume you want to program the HP E1411 multimeter for 10 DC voltage measurements. To specify 10 measurements you must type in the necessary command since the command is not on the multimeter menu. After selecting the VOLTMR menu, type the following commands and press **Return** after each command.

```
CONF:VOLT:DC
SAMP:COUN 10
READ?
```

These commands configure the multimeter, specify 10 measurements, and display the readings on the terminal.

Editing

The screen editing keys (shown on the following page) allow you to edit user-entered data or commands. When editing, the screen is in insert mode. That is, typed characters will be inserted into the string at the present cursor position.













Note

The key labels shown are found on all HP terminals (except HP terminals supporting ANSI terminal protocol). See "Using Supported Terminals" for equivalent key functions on your terminal.

General Key Descriptions

This section explains the function of each of the terminal interface's menu, menu control, and editing keys. If a key is not functional in a particular situation, pressing that key does nothing except to cause a beep.

Menu and Menu Control Keys

	through		Label menu choices for corresponding function keys.
	→		Returns to the <i>Select an instrument</i> menu.
			Returns to the previous menu level within an instrument menu or escapes from an input prompt. When you reach the top of an instrument's menu, the PRV_MENU label disappears.
			The screen can show a maximum of five menu choices at a time. When there are more than five menu choices, function key f6 becomes labeled MORE. Press MORE to display the next group of choices. By repeatedly pressing MORE you can display all groups of choices. After you have displayed all groups of choices, pressing MORE again returns to the first group of choices.
	→		Recalls the last command entered from the terminal interface. After recalling a command, it can be edited or re-executed. You can recall from a stack of previously executed commands by repeatedly pressing RCL_PREV. When you reach the bottom of the stack (the last line in the buffer), pressing RCL_PREV does nothing except to cause a beep.
	→		Recalls commands in the opposite order to that of RCL_PREV. Pressing RCL_NEXT does nothing until you have pressed RCL_PREV at least twice.
	→		Recalls the last SCPI command generated by a menu operation. For example, reading the time using the menus (SYSTEM, TIME, READ) generates and executes the SCPI command SYST:TIME?. A recalled command can be executed by pressing the Return key. You can also edit a recalled command before you execute it.
			Performs the same function as PRV_MENU.

Editing Keys



(Right arrow key.) Moves the cursor one character space to the right while leaving characters intact.



(Left arrow key.) Moves the cursor one character space to the left while leaving characters intact.



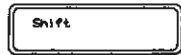
Erases the character at the present cursor position (for user-entered data only).



Erases the character to the left of the cursor (for user-entered data only).



(Clear-to-end key.) Erases all characters from the present cursor position to the end of the input line (for user-entered data only).



Selects the upper-case alphabetic characters or the character shown on the top half of a key.



Sets all alphabetic keys to uppercase (capitals); does not affect the other keys. To return to lowercase, press **Caps Lock** again.

Instrument Control Keys

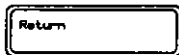


Resets only the selected instrument (equivalent of executing ***RST**). **RST_INST** also clears the instrument's terminal interface and remote input and output buffers. **RST_INST** is the only terminal interface key that can affect an instrument being operated from remote.



Clears the terminal interface input and output buffers (remote buffers are not cleared) of the selected instrument and returns to the top level of the instrument menu. Press **CLR_INST** whenever an instrument is busy, is not responding to terminal interface control, or to abort a command being entered from the terminal interface.

Other Keys



End of line. Enters your responses to menu prompts. Executes commands entered from the terminal keyboard (may be labeled **Enter** on your terminal emulator).



Selects alternate key definitions. These **CTRL** key sequences provide short-cuts to some of the menu sequences and also provide some functions not directly available from dedicated terminal keys. Some alternate key definitions are:

CTRL R = Instrument Reset

CTRL C = Clear Instrument

CTRL D = *Select an instrument* menu.

For a complete list of all **CTRL** Sequences, see Table 3-3 in this chapter. Users of the optional **IBASIC** interpreter should refer to their **IBASIC** manual set for additional editing functions.

Using Supported Terminals

The Display Terminal Interface supports several popular terminal brands and models. This chapter will show you how to access all of the terminal interface functions described previously using your supported terminal.

The Supported Terminals

The following list names the supported terminals and shows where to go for more information. If your terminal isn't named in this list, see "Using Other Terminals" in the next section.

- HP 700/92 Menu tutorial
- HP 700/94 Menu tutorial
- HP 700/22 See page 3-21
- HP 700/43 and WYSE WY-30[®] See page 3-23

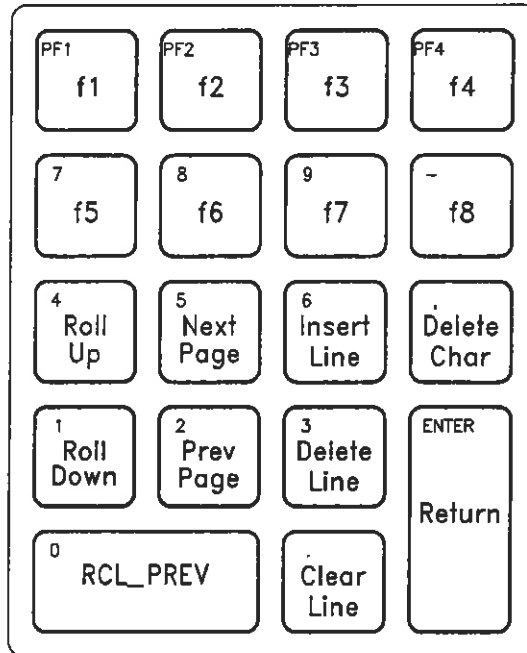
The keyboard guides provided for the listed terminals may be removed or copied, and placed near your keyboard while you go through the menu tutorial sections.

Using the HP 700/22

The HP 700/22 terminal emulates the DEC® VT100® or VT220® terminals. Some functions of the Display Terminal Interface have been mapped into keys with other labels. A keyboard map is provided for each of the emulation models. Use these keyboard maps to help locate the terminal interface functions.

VT100® Key Map

The symbols shown in the upper left corner of key each are now mapped with the function labeled in the center of each key.



E1405-DL VT100

Selecting VT100® Mode

To use the HP 700/22 in VT100® mode, press the Set-Up key and set the following configuration:

Fields	Value
Terminal Mode	EM100, 7 bit Ctrls
Columns	80
EM100 ID	EM100
Inhibit Auto Wrap	YES

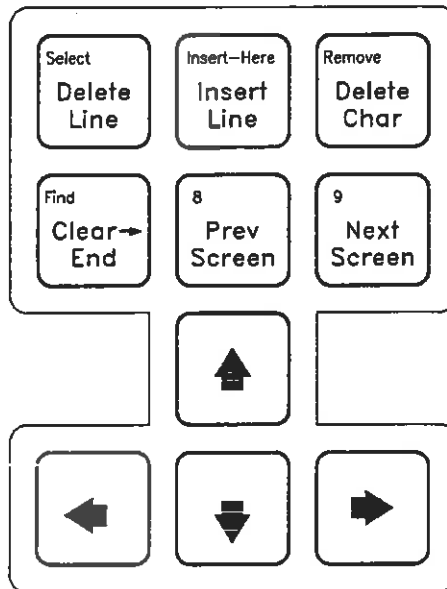
VT220® Key Map The function keys that are normally labeled f6 through f14 are now labeled:



Note

Because the HP 700/22 keyboard has nine function keys in the center of the keyboard, f4 is mapped twice

The symbols shown in the upper left corner of key each are now mapped with the function labeled in the center of each key.



E1405-DL VT220

Selecting VT220® Mode

To use the HP 700/22 in VT220® mode, press the Set-Up key and set the following configuration:

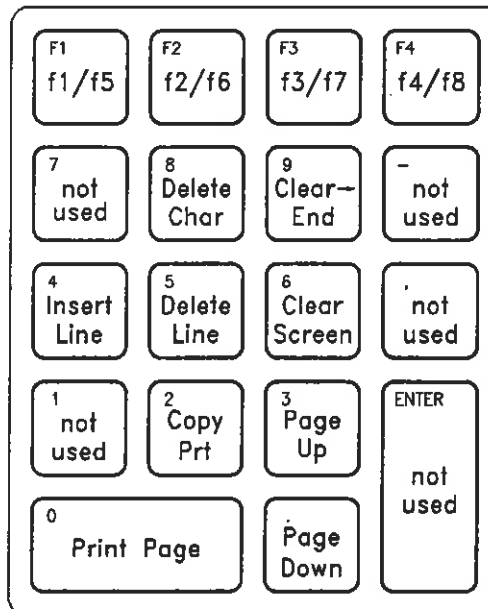
Fields	Value
Terminal Mode	EM200, 7 bit Ctrls
Columns	80
EM100 ID	EM220
Inhibit Auto Wrap	YES

Using the WYSE® WY-30™

With the WYSE® WY-30™ terminal, some functions of the Display Terminal Interface have been assigned to keys with other labels. Use this keyboard map to help locate these functions.

The symbols shown in the upper left corner of key each are now mapped with the function labeled in the center of each key.

Where two function key labels are shown, the one following the "/" character is accessed by pressing and holding the CTRL key while pressing the desired function key (e.g. to access the f6 function, press CTRL-f6).



E1405-DL WY30

Using Other Terminals

This section discusses using terminals which are not on the Supported Terminals list. Primarily this section is to help you use terminals which do not provide programmable soft keys (function keys). Without this capability, a terminal can not access the Display Terminal Interface's menus. Instead, the terminal interface provides a set of Terminal Interface Commands which allow you to select instruments by name or logical address. Once selected, you can type Common Commands or SCPI commands to the instrument. In addition, keyboard accessible control codes provide display control for terminals which may not have keys dedicated to those functions.

What "Not Supported" Means

Strictly speaking, a terminal is not supported if it has not been rigorously tested with the Display Terminal Interface. There are several HP terminals which may be compatible with the terminal interface. Terminals such as the DEC® VT100®, DEC® VT220®, and WYSE® WY-50™, or emulations of these may also work properly with the terminal interface. If you have one of these terminals, try it. Here is a list of terminals you should try.

HP 2392A

HP 2394A

DEC® VT100®

DEC® VT220®

WYSE® WY-50™

HP AdvanceLink terminal emulation software (configure as HP 2392A)

Testing Terminals for Compatibility

Here is how you test an unsupported terminal for compatibility with the Display Terminal Interface:

1. Connect your terminal and configure its communication parameters to match the mainframe's serial interface (see Appendix C)
2. With your terminal turned on and set to "remote mode", turn on the mainframe. After the mainframe power-on self-test, the display interface sends sequences of characters to your terminal which should cause it to return its identification. If the terminal ID matches one in a list kept by the terminal interface, it will send character sequences to program the function keys and their labels.
3. If you now see the "Select an instrument" prompt *and* the "Select an instrument" menu labels, your terminal is ready to try. Go to the beginning of this chapter and try the menus.
4. If you see only the "Select an instrument" prompt without the "Select an instrument" menu labels, your terminal did not return a recognized ID. To set the terminal type manually, type the Terminal Interface Command:

ST HP (followed by Return for HP terminals - 700/94, 700/92, 26xx,23xx)

or

ST HP70043 (followed by Return for the HP 700/43 terminal)

or

ST VT100 (followed by Return for VT100® emulators)

or

ST VT220 (followed by Return for VT220[®] emulators)

or

ST WYSE30 (followed by Return for WY-30[®] emulators)

or

ST WYSE50 (followed by Return for WY-50[®] emulators)

If you now see the "Select an instrument" menu labels, go to the beginning of this chapter and try the menus.

or

Turn the mainframe off and then on again.

Continue with this chapter to learn how to use your terminal without menus.

Using a Terminal Without Menus

You can still control instruments installed in your mainframe without using the terminal interface menus. In this case you will send Common Commands and SCPI commands to your instruments by typing them on your terminal keyboard, or through a computer interface.

Selecting Instruments

To send commands to, and receive responses from an instrument, you must first select that instrument. Two commands are provided to select instruments. They are; SI (Select Instrument), and SA (Select Address). These commands only work from the "Select an instrument" prompt. The commands can be typed in upper case or lower case.

SI SI selects an instrument by its name, exactly as it would appear in the "Select an instrument" menu (see Table 3-2). If your mainframe has more than one instrument with the same name, follow the name with a comma (,) and the desired instrument's logical address. Here are some examples of SI commands:

si voltmtr (selects a voltmeter instrument)

si switch (selects a switchbox instrument)

SI SWITCH (same as above)

si switch,16 (selects switchbox at logical address 16)

Table 3-2 Instrument Names for the SI Command

Menu Name	Instrument
SYSTEM	The System Instrument (built-in to the Command Module)
VOLTMTR	HP E1326A Standalone, or HP E1326A Scanning Voltmeter Modules
SWITCH	Switchbox composed of one or more HP Multiplexer Modules
DIG_I/O	HP E1330A Quad 8-Bit Digital Input/Output Module
IBASIC	Optional IBASIC interpreter
COUNTER	HP E1332A 4-Channel Counter/Totalizer, or HP E1333A Universal Counter Modules
D/A	HP E1328A Digital to Analog Converter Module

SA SA selects an instrument by its logical address. For multiple module instruments, use the logical address of the first module in the instrument. For example; SA 8 selects the instrument at logical address 8. When you have selected an instrument, the terminal interface will respond with an instrument prompt which is the instrument's menu name followed by its logical address (e.g. VOLTMTR_8:).

To get a list of the logical addresses used in your mainframe, send the SCPI command VXI:CONF:DLAD? to the System Instrument. Then to determine what instrument is at each logical address, send the command VXI:CONF:DLIS? n for each logical address in the list (where n is a logical address) .

Returning to the "Select an Instrument" Prompt

To return to the "Select an instrument" prompt, press and hold the CTRL key then press D.

Control Sequences for Terminal Interface Functions

The terminal interface provides the keyboard control sequences listed in Table 3-3. These can be thought of as keyboard short-cuts for compatible terminals (those which provide menu capability). Only those functions in the table which are shaded, operate for "UNKNOWN" terminal types (those which do not support menus). An "UNKNOWN" terminal type has very limited editing capability. It will not support the EDIT mode for the optional IBASIC interpreter. In the following table, † = IBASIC only,

Table 3-3 Control Sequence Functions

Terminal Key	Function	Control Sequence
Backspace	Deletes the character to the left of the cursor and moves cursor left.	CTRL-H
Del char	Delete character at the cursor position	CTRL-X
Clr →end	Clears line from cursor position to end of line	CTRL-L
Clear line	Clears line regardless of cursor position	CTRL-U
Insert line †	Inserts a blank line at the cursor position	CTRL-O
Delete line †	Deletes the line at the current cursor position	
End of line	Move cursor to the end of current line	CTRL-Z
Start of line	Move cursor to the beginning of current line	CTRL-A
Return	Terminates user entry	CTRL-M
RCL_MENU	Recalls the last command executed via the menu keys	CTRL-W
RCL_PREV	Recalls the last several commands executed via user input	CTRL-F
RCL_NEXT	After RCL_PREV, RCL_NEXT may be used to move forward through the recalled commands	CTRL-B
SEL_INST	Return to "Select an instrument" menu	CTRL-D
CLR_INST	Clear instrument's input and output buffers	CTRL-C
RST_INST	Like CLR_INST plus clears	CTRL-R

In Case of Difficulty

Problem:	Problem Cause/Solution:
<p>Error -113 undefined header error occurs after entering data in response to a menu prompt.</p>	<p>For some commands used by the menus, the data entered is appended to a command header. For example, if you enter "1" as the port number for a digital I/O module, the command used is DIG:HAND1:MODE NONE where HAND1 indicates the port number. If your entry was invalid or incorrect, error -113 occurs.</p>
<p>Following the power-on sequence or system reset the display shows:</p> <p>Configuration errors. Select SYSTEM Press any key to continue_</p>	<p>An unassigned device (incorrect logical address) was detected. If you cycle power or perform system reset, the display will show the logical address of the unassigned device. You can also check the logical addresses using the CONFIG? -- LADDS branch of the System Instrument menu. You can also use SYST:ERR? in the system instrument.</p>
<p>The display shows: "instrument in local lockout". Menus seem to work but nothing happens when I reach the bottom level or try to execute a command.</p>	<p>The terminal interface has been locked-out (HP-IB local lockout). You can re-enable menu operation by cancelling local lockout (from remote) or by cycling mainframe power.</p>
<p>Display cannot be removed from monitor mode.</p>	<p>Monitor mode was entered (DISP:MON:STAT ON command) and the terminal interface has also been locked out (HP-IB local lockout). Either cancel the local lockout or execute DISP:MON:STAT OFF (from remote).</p>
<p>Display shows:</p> <p>Can not connect to instrument Press any key to continue_</p>	<p>A hardware or software problem has occurred in the instrument preventing it from responding to terminal interface control.</p>
<p>After selecting an instrument the display shows:</p> <p>"busy".</p>	<p>The instrument is busy performing an operation. Press Clear Instr to abort the instrument operations and allow the terminal interface to access the instrument.</p>
<p>Display shows:</p> <p>Instrument in use by another display. Press any key to continue_</p>	<p>The instrument has already been selected from another terminal interface. An instrument can only be "attached" to one display at a time. At the other terminal interface, press Select Instr. The instrument can now be selected from the desired terminal interface.</p>

Notes

Instrument Menus

This section contains charts showing the structure and content for all terminal interface instrument menus. The SCPI or Common Commands used and descriptions of menu-controlled instrument operations are also included in the charts.

System Instrument Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	User Entry	Command(s) Used	Description
SYSTEM	CONFIG?	LADDRS				logical address	VXI:CONF:DLAD?	Displays logical addresses of mainframe instruments
		DEVICE					VXI:CONF:DLIS? <log_addr>	Displays information about the device at the specified logical address. (Refer to the Command Reference for details)
	HP:IB?						SYST:COMM:CF:IB:ADDR?	Displays HP-IB address
	RS232	BAUD	READ	SET	300	card number	SYST:COMM:SER[n]:BAUD?	Read current baud rate
					1200	card number	SYST:COMM:SER[n]:BAUD 300	Sets the serial interface baud rate to 300
					2400	card number	SYST:COMM:SER[n]:BAUD 1200	Sets the serial interface baud rate to 1200
					9600	card number	SYST:COMM:SER[n]:BAUD 2400	Sets the serial interface baud rate to 2400
					19200	card number	SYST:COMM:SER[n]:BAUD 9600	Sets the serial interface baud rate to 9600
	PARITY	READ	SET	EVEN	card number	SYST:COMM:SER[n]:PAR?	SYST:COMM:SER[n]:BAUD 19200	Sets the serial interface baud rate to 19200
				ODD	card number	SYST:COMM:SER[n]:PAR EVEN	Read current parity type	
				ONE	card number	SYST:COMM:SER[n]:PAR ODD	Sets the serial interface parity to even	
				ZERO	card number	SYST:COMM:SER[n]:PAR ODD	Sets the serial interface parity to odd	
				NONE	card number	SYST:COMM:SER[n]:PAR ONE	Sets the serial interface parity to one	
	BITS	READ	SET	7	card number	SYST:COMM:SER[n]:PAR ZERO	SYST:COMM:SER[n]:PAR NONE	Sets the serial interface parity to zero
				8	card number	SYST:COMM:SER[n]:PAR ZERO	SYST:COMM:SER[n]:PAR NONE	Sets the serial interface parity to none
	PACE	READ	SET	XON/OFF	card number	SYST:COMM:SER[n]:BITS?	SYST:COMM:SER[n]:BITS 7	Read current data bit width
				NONE	card number	SYST:COMM:SER[n]:BITS 7	SYST:COMM:SER[n]:BITS 8	Sets the data width to 7 bits
	XON/OFF	READ	SET		card number	SYST:COMM:SER[n]:BITS 8	SYST:COMM:SER[n]:PACE?	Sets the data width to 8 bits
					card number	SYST:COMM:SER[n]:PACE?	SYST:COMM:SER[n]:PACE XON	Read current pacing type
					card number	SYST:COMM:SER[n]:PACE XON	SYST:COMM:SER[n]:PACE NONE	Enables XON/ XOFF software handshaking
					card number	SYST:COMM:SER[n]:PACE NONE	SYST:COMM:SER[n]:PACE NONE	Disables XON/ XOFF software handshaking

(continued on following page)

System Instrument Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	User Entry	Command(s) Used	Description	
(continued from previous page)	CONTROL	DTR	READ	SET	ON	card number	SYST:COMM:SER[n]:CONT:DTR?	Read current setting for DTR line	
			READ	SET	OFF	card number	SYST:COMM:SER[n]:CONT:DTR ON	Set DTR line to static +V	
		KTS	READ	SET	IBFULL	card number	SYST:COMM:SER[n]:CONT:DTR OFF	Set DTR line to static -V	
			READ	SET	STANDARD	card number	SYST:COMM:SER[n]:CONT:DTR IBF	Set DTR for hardware handshaking	
			READ	SET	ON	card number	SYST:COMM:SER[n]:CONT:DTR STAN	DTR operates to RS-232 standard	
			READ	SET	OFF	card number	SYST:COMM:SER[n]:CONT:KIS?	Read current setting for KTS line	
	STORE	READ	SET	IBFULL	card number	SYST:COMM:SER[n]:CONT:KIS ON	Set KTS line to static +V		
		READ	SET	STANDARD	card number	SYST:COMM:SER[n]:CONT:KIS OFF	Set KTS line to static -V		
		READ	SET	ON	card number	SYST:COMM:SER[n]:CONT:KTS IBF	Set KTS for hardware handshaking		
	DEBUG	READ	WRITE	MESSAGE	COMMAND	QUERY	card number	SYST:COMM:SER[n]:CONT:KTS STAN	KTS operates to RS-232 standard
				MESSAGE	COMMAND	QUERY	card number	DIAG:COMM:SER[n]:STORE	Store current serial communications settings into non-volatile storage.
				MESSAGE	COMMAND	QUERY	card number	VXI:READ? <laddr>, <reg>	Read register in A16 address space.
		SEND	MESSAGE	RECEIVE	COMMAND	QUERY	laddr, reg_num	VXI:WRIT <laddr>, <reg>, <data>	Write data to register in A16 address space.
				RECEIVE	COMMAND	QUERY	laddr, string	VXI:SEND <laddr>, <string>	Send SCPI command to message-based instrument at laddr
				RECEIVE	COMMAND	QUERY	laddr,command	VXI:SEND:COMM <laddr>, <command>	Send word serial command to laddr
RECEIVE		MESSAGE	RECEIVE	COMMAND	QUERY	laddr, query	VXI:SEND:COMM? <laddr>, <query>	Send word serial command and wait for response.	
			RECEIVE	COMMAND	QUERY	laddr	VXI:REC? <laddr>	Receive message from message-based device.	
			RECEIVE	COMMAND	QUERY	laddr	VXI:RES <laddr>	Soft reset of device at laddr.	
RECEIVE	MESSAGE	RECEIVE	COMMAND	QUERY	laddr	VXI:QUER? <laddr>	Read Data Low register		
		RECEIVE	COMMAND	QUERY	laddr				
		RECEIVE	COMMAND	QUERY	laddr				

(continued on following page)

Switchbox Menu

Menu Levels and Content

Level 1	Level 2	Level 3	User Entry	Command(s) Used	Description
SWITCH	MONITOR		card number ‡ or AUTO	DISP:MON:CARD <card_number>;STAT ON	Monitor instrument operations
	OPEN		channel list †	OPEN (@channel_list)	Open channel(s)
	CLOSE		channel list †	CLOS (@channel_list)	Close channel(s)
	SCAN	SET_UP	channel list †	TRIG:SOUR HOLD;SCAN <channel_list>;INIT	Set up channels to scan
		STEP	channel list †	TRIG	Step to next channel in scan list
	CARD	TYPE?	card number ‡	SYST:CTYP? <card_number>	Display module ID information
		DESCR?	card number ‡	SYST:CDES? <card_number>	Display module description
		RESET	card number ‡	SYST:CPON <card_number>	Return module to power-on state
	TEST			*TST?	Runs self-test, displays results (+0 =pass; any other number =fail)

† Channel lists are of the form "ccnn" (single channel), "ccnn,ccnn" (two or more channels) or "ccnn:ccnn" (range of channels); where "cc" is the card number and "nn" is the channel number. For example, to access channel 2 on card number 1 specify 102.

‡ The card number identifies a module within the Switchbox. The switch module with the lowest logical address is always card number 01. The switch module with the next successive logical address is card number 02 and so on.

Notes

Scanning Voltmeter Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	User Entry	Command(s) Used	Description
VOLTMTR	MONITOR			channel list † or 0 for auto	DISP: MON: CHAN <channel_list>; STAT ON	Monitor instrument operations
	VDC			channel list †	MEAS: VOLT: DC? <channel_list >	Measure DC voltage on each channel
	VAC			channel list †	MEAS: VOLT: AC? <channel_list >	Measure AC voltage on each channel
	OHM			channel list †	MEAS: RES? <channel_list >	Measure 2-wire resistance on each channel
	TEMP	TCOUPLE	B	channel list †	MEAS: TEMP? TC,B, <channel_list >	Measure °C of B thermocouple on each channel
			E	channel list †	MEAS: TEMP? TC,E, <channel_list >	Measure °C of E thermocouple on each channel
			J	channel list †	MEAS: TEMP? TC,J, <channel_list >	Measure °C of J thermocouple on each channel
			K	channel list †	MEAS: TEMP? TC,K, <channel_list >	Measure °C of K thermocouple on each channel
			NI14	channel list †	MEAS: TEMP? TC,NI14, <channel_list >	Measure °C of NI14 thermocouple on each channel
			N28	channel list †	MEAS: TEMP? TC,N28, <channel_list >	Measure °C of N28 thermocouple on each channel
			R	channel list †	MEAS: TEMP? TC,R, <channel_list >	Measure °C of R thermocouple on each channel
			S	channel list †	MEAS: TEMP? TC,S, <channel_list >	Measure °C of S thermocouple on each channel
			T	channel list †	MEAS: TEMP? TC,T, <channel_list >	Measure °C of T thermocouple on each channel
		THERMIS	2252	channel list †	MEAS: TEMP? THER,2252, <channel_list >	Measure °C of 2252 Ω thermistor on each channel
			5K	channel list †	MEAS: TEMP? THER,5000, <channel_list >	Measure °C of 5k Ω thermistor on each channel
			10K	channel list †	MEAS: TEMP? THER,10000, <channel_list >	Measure °C of 10k Ω thermistor on each channel
		RTD	385	channel list †	MEAS: TEMP? RTD,85, <channel_list >	Measure °C of 385 RTD on each channel (4-wire)
			392	channel list †	MEAS: TEMP? RTD,92, <channel_list >	Measure °C of 392 RTD on each channel (4-wire)
		STRAIN	QUARTER	channel list †	MEAS: STR: QJAR? <channel_list >	Measure strain with quarter bridge
			HALF	channel list †	MEAS: STR: HBEN? <channel_list >	Measure strain with bending half bridge
			POISSON	channel list †	MEAS: STR: HPO? <channel_list >	Measure strain with Poisson half bridge
			FULL	channel list †	MEAS: STR: FBEN? <channel_list >	Measure strain with bending full bridge
			BENPOIS	channel list †	MEAS: STR: FBP? <channel_list >	Measure strain with Bending Poisson full bridge
			POISSON	channel list †	MEAS: STR: FPO? <channel_list >	Measure strain with Poisson full bridge

(continued on following page)

Scanning Voltmeter Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	User Entry	Command(s) Used	Description
(continued from previous page)						
		UNSTRN		channel list †	MEAS:STR:UNST? <channel_list >	Measure bridge unstrained
		DIAG	COMPRES	channel list †	MEAS:STR:QOOM? <channel_list >	Compression shunt diagnostic
			TENSION	channel list †	MEAS:STR:QIEN? <channel_list >	Tension shunt diagnostic
	CARD	TYPE?		card number ‡	SYST:CIYP? <card_number >	Displays module ID information
		DESCR?		card number ‡	SYST:COES? <card_number >	Displays module description
			TEST		*TST?	Runs self-test, displays results (+0 =pass; any other number =fail)

† Channel lists are of the form "ccnn" (single channel), "ccnn,ccnn" (two or more channels) or "ccnn:ccnn" (range of channels); where "cc" is the card number and "nn" is the channel number. For example, to access channel 2 on card number 1 specify 102.

‡ The card number identifies a module within the Switchbox. The switch module with the lowest logical address is always card number 01. The switch module with the next successive logical address is card number 02 and so on.

HP E1326B/E1411B 5 1/2 Digit Multimeter (Standalone) Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	User Entry	Command(s) Used	Description
VOLTMTR	MONITOR					Display instrument operations
	VDC				DISP: MON: STAT ON	Measure DC volts
	VAC				MEAS: VOLT: DC?	Measure AC volts
	OHM				MEAS: VOLT: AC?	Measure 4-wire ohms
	TEMP	THERMIS	2.252		MEAS: FRES?	Measure °C of 2252Ω thermistor (4-wire measurement)
			5K		MEAS: TEMP? FTH, 2252	Measure °C of 5kΩ thermistor (4-wire measurement)
			10K		MEAS: TEMP? FTH, 5000	Measure °C of 10kΩ thermistor (4-wire measurement)
		RID	385		MEAS: TEMP? FTH, 10000	Measure °C of 100Ω RID with alpha = 385 (4-wire measurement)
			392		MEAS: TEMP FRID, 85?	Measure °C of 100Ω RID with alpha = 392 (4-wire measurement)
	TEST				MEAS: TEMP FRID, 92?	Run self-test, display results (0 = pass; any other number = fail)
					*TST?	

† Channel lists are of the form "ccnn" (single channel), "ccnn,ccnn" (two or more channels) or "ccnn:ccnn" (range of channels); where "cc" is the card number and "nn" is the channel number. For example, to access channel 2 on card number 1 specify 102.

‡ The card number identifies a module within the Switchbox. The switch module with the lowest logical address is always card number 01. The switch module with the next successive logical address is card number 02 and so on.

HP E1328A 4-Channel D/A Converter Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	User Entry	Command(s) Used	Description	
D/A	MONITOR	CHAN1			DISP: MON: CHAN 1; STAT ON	Monitor instrument operations on channel 1	
		CHAN2			DISP: MON: CHAN 2; STAT ON	Monitor instrument operations on channel 2	
		CHAN3			DISP: MON: CHAN 3; STAT ON	Monitor instrument operations on channel 3	
		CHAN4			DISP: MON: CHAN 4; STAT ON	Monitor instrument operations on channel 4	
	OUTPUT	AUTO				DISP: MON: CHAN AUTO; STAT ON	Monitor instrument operations on active channel
		VOLTAGE	CHAN1		voltage †	VOLT1 <voltage >	Output voltage on channel 1
			CHAN2		voltage †	VOLT2 <voltage >	Output voltage on channel 2
			CHAN3		voltage †	VOLT3 <voltage >	Output voltage on channel 3
			CHAN4		voltage †	VOLT4 <voltage >	Output voltage on channel 4
		CURRENT	CHAN1		current ‡	CURR1 <current >	Output current on channel 1
			CHAN2		current ‡	CURR2 <current >	Output current on channel 2
			CHAN3		current ‡	CURR3 <current >	Output current on channel 3
	CHAN4			current ‡	CURR4 <current >	Output current on channel 4	
	TEST				*TST?	Run self-test, display results (+0 =pass; any other number =fail)	

†Enter voltage values in volts. Typical examples are: +3.5, -.2, +500E-3.

‡Enter current values in amps. Typical examples are: .05, +200E-3.

HP E1330A Quad 8-Bit Digital Input/Output Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	User Entry	Command(s) Used	Description	
DIG_I/O	MONITOR	PORT0			DISP:MON:CHAN 0;STAT ON	Monitor instrument operations on port 0	
		PORT1			DISP:MON:CHAN 1;STAT ON	Monitor instrument operations on port 1	
		PORT2			DISP:MON:CHAN 2;STAT ON	Monitor instrument operations on port 2	
		PORT3			DISP:MON:CHAN 3;STAT ON	Monitor instrument operations on port 3	
		AUTO			DISP:MON:CHAN AUTO;STAT ON	Monitor instrument operations on any active port	
	READ	R_BYTE	PORT0			DIG:HAND0:MODE NONE;MEAS: DIG:DATA0?	Reads port 0 after handshake
			PORT1			DIG:HAND1:MODE NONE;MEAS: DIG:DATA1?	Reads port 1 after handshake
			PORT2			DIG:HAND2:MODE NONE;MEAS: DIG:DATA2?	Reads port 2 after handshake
		R_BIT	PORT3			DIG:HAND3:MODE NONE;MEAS: DIG:DATA3?	Reads port 3 after handshake
			PORT0	bit (0-7)		DIG:HAND0:MODE NONE;MEAS: DIG:DATA0:BITm?	Reads bit m on port 0 after handshake
			PORT1	bit (0-7)		DIG:HAND1:MODE NONE;MEAS: DIG:DATA1:BITm?	Reads bit m on port 1 after handshake
	WRITE	W_BYTE	PORT2			DIG:HAND2:MODE NONE;MEAS: DIG:DATA2:BITm?	Writes data to port 2
			PORT3			DIG:HAND3:MODE NONE;MEAS: DIG:DATA3:BITm?	Writes data to port 3
		W_BIT	PORT0	data (0-255)		DIG:HAND0:MODE NONE;MEAS: DIG:DATA0 <data >	Writes data to bit m on port 0
			PORT1	data (0-255)		DIG:HAND1:MODE NONE;MEAS: DIG:DATA1 <data >	Writes data to bit m on port 1
		PORT2	data (0-255)		DIG:HAND2:MODE NONE;MEAS: DIG:DATA2 <data >	Writes data to bit m on port 2	
		PORT3	data (0-255)		DIG:HAND3:MODE NONE;MEAS: DIG:DATA3 <data >	Writes data to bit m on port 3	
		PORT0	bit (0-7), value (0,1)		DIG:HAND0:MODE NONE;MEAS: DIG:DATA0:BITm <value >	Writes data to bit m on port 0	
		PORT1	bit (0-7), value (0,1)		DIG:HAND1:MODE NONE;MEAS: DIG:DATA1:BITm <value >	Writes data to bit m on port 1	
		PORT2	bit (0-7), value (0,1)		DIG:HAND2:MODE NONE;MEAS: DIG:DATA2:BITm <value >	Writes data to bit m on port 2	
		PORT3	bit (0-7), value (0,1)		DIG:HAND3:MODE NONE;MEAS: DIG:DATA3:BITm <value >	Writes data to bit m on port 3	

Notes

HP E1332A 4-Channel Counter/Totalizer Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	Level 5	User Entry	Command(s) Used	Description
COUNTER	MONITOR	CHAN1				DISP: MON: CHAN 1; STAT ON	Monitor instrument operations on channel 1
		CHAN2				DISP: MON: CHAN 2; STAT ON	Monitor instrument operations on channel 2
		CHAN3				DISP: MON: CHAN 3; STAT ON	Monitor instrument operations on channel 3
		CHAN4				DISP: MON: CHAN 4; STAT ON	Monitor instrument operations on channel 4
		AUTO				DISP: MON: CHAN AUTO; STAT ON	Monitor instrument operations on active channel
	INPUT	LEVEL	CHAN1 &2		voltage †	SENS1: EVEN: LEV <value >	Set level trigger voltage for channels 1 & 2
			CHAN3 &4		voltage †	SENS3: EVEN: LEV <value >	Set level trigger voltage for channels 3 & 4
		SLOPE	CHAN1	POS		SENS1: EVEN: SLOP POS	Positive level trigger slope for channel 1
				NEG		SENS1: EVEN: SLOP NEG	Negative level trigger slope for channel 1
			CHAN2	POS		SENS2: EVEN: SLOP POS	Positive level trigger slope for channel 2
				NEG		SENS2: EVEN: SLOP NEG	Negative level trigger slope for channel 2
			CHAN3	POS		SENS3: EVEN: SLOP POS	Positive level trigger slope for channel 3
				NEG		SENS3: EVEN: SLOP NEG	Negative level trigger slope for channel 3
			CHAN4	POS		SENS4: EVEN: SLOP POS	Positive level trigger slope for channel 4
				NEG		SENS4: EVEN: SLOP NEG	Negative level trigger slope for channel 4
		ISOLATE		ON		INP: ISOL ON	Input isolation on
				OFF		INP: ISOL OFF	Input isolation off
		FILTER		ON		INP: FILT ON	Input filter on
				OFF		INP: FILT OFF	Input filter off
				FREQ	frequency ‡	INP: FILT: FREQ <value >	Set input filter frequency
	FREQ		CHAN1			TRIG: SOUR IMM;: MEAS1: FREQ?	Frequency measurement on channel 1
			CHAN3			TRIG: SOUR IMM;: MEAS3: FREQ?	Frequency measurement on channel 3
	PERIOD		CHAN1			TRIG: SOUR IMM;: MEAS1: PER?	Period measurement on channel 1
			CHAN3			TRIG: SOUR IMM;: MEAS3: PER?	Period measurement on channel 3

(continued on following page)

HP E1333A 3-Channel Universal Counter Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	Level 5	User Entry	Command(s) Used	Description	
COUNTER	MONITOR	CHAN1					DISP: MON: CHAN 1; STAT ON	Monitor instrument operations on channel 1
		CHAN2					DISP: MON: CHAN 2; STAT ON	Monitor instrument operations on channel 2
		CHAN3					DISP: MON: CHAN 3; STAT ON	Monitor instrument operation on channel 3
		AUTO					DISP: MON: CHAN AUTO; STAT ON	Monitor instrument operations on active channel
	INPUT	LEVEL	CHAN1			voltage †	SENS1: EVEN: LEV <value >	Set trigger level voltage for channel 1
			CHAN2			voltage †	SENS2: EVEN: LEV <value >	Set trigger level voltage for channel 2
		SLOPE	CHAN1	POS			SENS1: EVEN: SLOP POS	Positive trigger slope for channel 1
			CHAN2	NEG			SENS1: EVEN: SLOP NEG	Negative trigger slope for channel 1
		COUPLE	AC				SENS2: EVEN: SLOP POS	Positive trigger slope for channel 2
			DC				SENS2: EVEN: SLOP NEG	Negative trigger slope for channel 2
	IMPED	50_OHM					INP: COUP AC	AC-coupled input (channels 1 & 2 only)
							INP: COUP DC	DC-coupled input (channels 1&2)
		1_MOHM					INP: IMP 50	50Ω input resistance (channels 1 & 2 only)
	ATTEN	0dB					INP: IMP 1e6	1MΩ input resistance (channels 1 & 2 only)
		20dB					INP: ATT 0	No input attenuation (channels 1 & 2 only)
FILTER	ON					INP: ATT 20	20dB input attenuation (channels 1 & 2 only)	
		OFF				INP: FILT ON	Input filter on (channels 1 & 2 only)	
FREQ	CHAN1					INP: FILT OFF	Input filter off (channels 1 & 2 only)	
						TRIG: SOUR IMM; MEAS1: FREQ?	Frequency measurement on channel 1	
						TRIG: SOUR IMM; MEAS2: FREQ?	Frequency measurement on channel 2	
PERIOD	CHAN1					TRIG: SOUR IMM; MEAS3: FREQ?	Frequency measurement on channel 3	
		CHAN2				TRIG: SOUR IMM; MEAS1: PER?	Period measurement on channel 1	
						TRIG: SOUR IMM; MEAS2: PER?	Period measurement on channel 2	

(continued on following page)

HP E1333A 3-Channel Universal Counter Menu

Menu Levels and Content

Level 1	Level 2	Level 3	Level 4	Level 5	User Entry	Command(s) Used	Description
(continued from previous page)	TIMENT	CHAN1 CHAN2				TRIG:SOUR IMM;; MEAS1:TINT?	Time interval measurement on channel 1
	POS_PW	CHAN1 CHAN2				TRIG:SOUR IMM;; MEAS2:TINT?	Time interval measurement on channel 2
	NEG_PW	CHAN1 CHAN2				TRIG:SOUR IMM;; MEAS1:PWID?	Positive pulse width measurement on channel 1
						TRIG:SOUR IMM;; MEAS2:PWID?	Positive pulse width measurement on channel 2
						TRIG:SOUR IMM;; MEAS1:NWID?	Negative pulse width measurement on channel 1
						TRIG:SOUR IMM;; MEAS2:NWID?	Negative pulse width measurement on channel 2
	RATIO	CHAN1 CHAN2				TRIG:SOUR IMM;; MEAS1:RAT?	Ratio of channel 1/channel 2
						TRIG:SOUR IMM;; MEAS2:RAT?	Ratio of channel 2/channel 1
	TOTALIZ	CHAN1 CHAN2				TRIG:SOUR IMM;; CONF1:TOT;:INIT1 FEIC1?	Totalize on channel 1 Display totalize count
						TRIG:SOUR IMM;; CONF2:TOT;:INIT2 FEIC2?	Totalize on channel 2 Display totalize count
	TEST					*IST?	Run self-test, display results (+0 =pass; any other number =fail)

†Enter voltage values in volts. Typical examples are: +3.5, -2, +500E-3.

Status and Interrupts

About this Chapter

This chapter describes the status system structure used by the System Instrument (Command Module) and other SCPI (Standard Commands for Programmable Instruments) devices. Included is information on how to monitor instrument status, interrupt the computer, and synchronize one or more instruments to an external computer. The discussion and examples apply to instruments controlled from a computer over the HP-IB.

Command references for the supported IEEE 488.2 Common Commands and IEEE 488.1 HP-IB Messages are located near the end of this chapter. This chapter contains the following sections:

- Status System Structure 4-2
- Clearing Status 4-10
- Interrupting an External Computer 4-10
- Synchronizing an External Computer and Instruments 4-12

NOTE

Examples that require showing a computer language are written for HP 9000 Series 200/300 Computers using HP BASIC language.

Status System Structure

The instrument status structure monitors important events for an instrument such as when an error occurs or when a reading is available. All instruments have the following status groups and registers within those groups:

- Status Byte Status Group
 - status byte register
 - service request enable register
- Standard Event Status Group
 - standard event status register
 - standard event status enable register
- Operation Status Group
 - condition register
 - event register
 - enable register
- Questionable Data Status Group
 - condition register
 - event register
 - enable register

You read and configure the registers in the Status Byte and Standard Event groups using Common Commands. These are the most commonly used instrument registers. The registers in the Standard Operation Status group and Questionable Data status group are configured using the commands in the STATus subsystem.

NOTE

The Status Byte, Standard Event, and Operation Status groups are the only groups covered in this chapter. The Questionable Data status group is supported by the system instrument (Command Module) but is not used by the system instrument. Commands affecting this status group (Chapter 5) are accepted but have no effect.

Refer to the STATus subsystem in the Command Reference of the individual plug-in module manuals to determine how a module uses the Operation Status group and Questionable Data status groups. If the STAT:OPER or STAT:QUES commands are not documented in the plug-in module manual, that module does not use the registers.

The Status Byte Register

As shown in Figure 4-1, the Status Byte register is the highest-level register in the status structure. This register contains bits which summarize information from the other status groups.

NOTE

The bits in the other status group registers must be specifically enabled to be reported in the Status Byte register. Refer to "Unmasking Standard Event Status Bits" (later in this chapter) for more information.

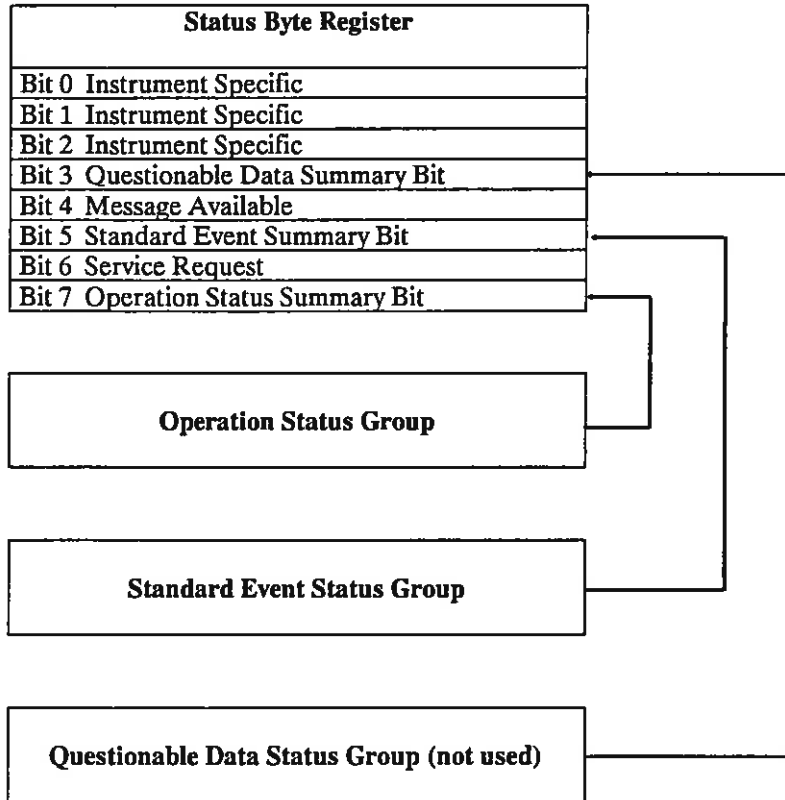


Figure 4-1. Status Structure

Table 4-1 shows each of the Status Byte register bits and describes the event that will set each bit.

Table 4-1. Status Byte Register

Bit Number	Decimal Weight	Description
0	1	Instrument Specific (not used by most instruments)
1	2	Instrument Specific (not used by most instruments)
2	4	Instrument Specific (not used by most instruments)
3	8	Questionable Data Status Group Summary Bit. One or more events in the Questionable Data Status group have occurred and set bit(s) in those registers.
4	16	Message Available. The instrument's output queue contains information. This bit can be used to synchronize data exchange with an external computer. For example, you can send a query command to the instrument and then wait for this bit to be set. The HP-IB is then available for other use while the program is waiting for the instrument to respond.
5	32	Standard Event Status Group Summary Bit. One or more enabled events in the Standard Event Status Register have occurred and set bit(s) in that register.
6	64	Service Request--Service is requested by the instrument and the HP-IB SRQ line is set true. This bit will be set when any other bit of the Status Byte Register is set and has been enable to assert SRQ by the *SRE command.
7	128	Operation Status Group Summary Bit. One or more events in the Operation Status Group have occurred and set bit(s) in those registers.

Reading the Status Byte Register

You can read the Status Byte register using either the *STB? command or an HP-IB serial poll. Both methods return the decimal weighted sum of all set bits in the register. The difference between the two methods is that *STB? does not clear bit 6 (Service Request); serial poll does clear bit 6. No other status register bits are cleared by either method with the exception of the Message Available bit (bit 4) which may be cleared as a result of reading the response to *STB?. In addition, using an HP-IB serial poll lets you read the status byte without interrupting the instrument parser. The *STB? method requires the instrument to process the command. This can generate interrupt query errors if the instrument is executing another query.

The following program uses the *STB? command to read the contents of the system instrument's (Command Module's) Status Byte register.

```

10 OUTPUT 70900;"*STB?"           Read Status Byte Register
20 ENTER 70900; A                 Enter weighted sum
30 PRINT A                         Print weighted sum
40 END

```

For example, assume bit 3 (weight = 8) and bit 7 (weight = 128) are set. The above program returns the sum of the two weights (136).

The following program reads the system instrument's Status Byte register using the HP-IB Serial Poll command.

```
10 P = SPOLL(70900)
```

*Read Status Byte Register using
Serial Poll, place weighted sum
in P*

```
20 PRINT P
```

Print weighted sum

```
30 END
```

Service Request Enable Register

The Service Request Enable register is used to "unmask" bits in the Status Byte register. When an unmasked Status Byte register bit is set to '1', a service request is sent to the computer over HP-IB.

The command used to unmask Status Byte register bits is:

```
*SRE <mask >
```

where *<mask >* is the decimal weight of the bit to be unmasked, or is the sum of the decimal weights if multiple bits are to be unmasked. For example, executing:

```
*SRE 16
```

unmasks the *message available (MAV)* bit in the Status Byte register. Sending:

```
*SRE 48
```

unmasks the *message available (MAV)* and *event status bit (ESB)*.

You can determine which bits in the Status Byte register are unmasked by sending the command:

```
*SRE?
```

This command returns the decimal weighted sum of all unmasked bits.

The Service Request Bit

Note that the Service Request bit (bit 6) in the Status Byte register does not have a mask. Bit 6 is set any time another Status Byte register bit is set. If the other bit which is set is unmasked, a service request is generated.

Clearing the Service Request Enable Register

The Service Request Enable register mask is cleared (each bit masked except bit 6) by sending the command:

```
*SRE 0
```

If *PSC 1 has been executed, the Service Request Enable register mask is cleared when power is cycled. If *PSC 0 has been executed, the mask is unchanged when power is cycled. (*PSC? queries the setting.)

Standard Event Status Register

The Standard Event Status Register in the Standard Event status group monitors the instrument status events shown in Table 4-2. When one of these events occurs, it sets a corresponding bit in the Standard Event Status Register.

NOTE

The Standard Event Status Register bits are not reported in the Status Byte Register unless unmasked by the Standard Event Status Enable Register. Refer to the section "Unmasking Standard Event Status Bits" for more information.

Table 4-2. Standard Event Status Register

Bit Number	Decimal Weight	Description
0	1	Operation Complete. The instrument has completed all pending operations. This bit is set in response to the *OPC command.
1	2	Request Control. An instrument is requesting permission to become the active HP-IB controller.
2	4	Query Error. A problem has occurred in the instrument's output queue.
3	8	Device Dependent Error. An instrument operation did not complete possibly because of an abnormal hardware or firmware condition (overload occurred, self-test failure, loss of calibration or configuration memory, etc.)
4	16	Execution Error. The instrument cannot do the operation(s) requested by a command.
5	32	Command Error. The instrument cannot understand or execute the command.
6	64	User Request. The instrument is under local (front panel) control.
7	128	Power-On. Power has been applied to the instrument. You must execute the *PSC 0 command to the System Instrument to allow this bit to remain enabled when power is cycled. See the *PSC command later in this chapter for an example.
8-15		Reserved for future use (always return zero).

Unmasking Standard Event Status Bits

To allow any of the Standard Event Status register bits to set bit 5 (ESB) of the Status Byte register, you must first unmask the bit(s) using the Standard Event Status Enable register with the command:

*ESE

For example, suppose your application requires an interrupt whenever any type of error occurs. The error related bits in the Standard Event Status register are bits 2 through 5. The sum of the decimal weights of these bits is 60. You can enable any one of these bits to set bit 5 in the Status Byte Register by sending:

*ESE 60

If you want to generate a service request following any one of these errors, you can do so by unmasking bit 5 (ESB) in the Status Byte register:

*SRE 32

*ESE 60

Now, whenever an error occurs, it will set one of the bits 2 - 5 in the Standard Event Status register which will set bit 5 in the Status Byte register. Since bit 5 is

unmasked, an HP-IB service request (SRQ) will be generated. ("Interrupting the External Computer", later in this chapter contains an example program which demonstrates this sequence).

Note that the Standard Event Status Register bits that are not unmasked still respond to their corresponding conditions. They do not, however, set bit 5 in the Status Byte Register.

Reading the Standard Event Status Enable Register Mask

You can determine which bits in the Standard Event Status register are unmasked with the command:

***ESE?**

This command returns the decimal weighted sum of all unmasked bits.

The Standard Event Status Enable register is cleared (all bits masked) by sending the command:

***ESE 0**

Reading the Standard Event Status Register

You can determine which bits in the Standard Event Status register are set using the command:

***ESR?**

This command returns the decimal weighted sum of all set bits. *ESR? clears the register. *CLS also clears the register.

Both of these commands return the decimal weighted sum of all set or enabled bits.

Operation Status Group

The registers in the Standard Operation Status Group provide information about the state of measurement functions within an instrument. These functions are represented by bits in the Condition register which is described in Table 4-3.

The System Instrument (Command Module) only uses bit 8 in the Condition register. Bit 8 (when set) indicates that an interrupt set up by the DIAGnostic:INTerrupt commands has occurred and has been acknowledged.

NOTE

The registers in the Operation Status Group and the DIAGnostic:INTerrupt commands are only used when, for a specific VXIbus interrupt line, it is necessary to replace the operating system's interrupt service routine with the System Instrument's service routine. Hewlett-Packard VXIbus devices used with the Command Module use the operating system service routine. The VXIbus interrupt line that is used by these devices (primarily line 1), should not be used with the DIAGnostic:INTerrupt commands.

The DIAGnostic:INTerrupt commands are covered in Chapter 5.

Table 4-3. Operation Status Group - Condition Register

Bit Number	Decimal Weight	Description
0	1	Calibrating
1	2	Settling
2	4	Ranging
3	8	Sweeping
4	16	Measuring
5	32	Waiting for TRG
6	64	Waiting for ARM
7	128	Correcting
8	256	Interrupt acknowledged (System Instrument)
9-12		Instrument Dependent
13-14		Reserved
15		Always zero

Reading the Condition Register

When an event monitored by the Condition register has occurred or is occurring, a corresponding bit in the register is set. The bit which is set can be determined with the command:

STATUS:OPERATION:CONDITION?

The data which is returned is the decimal weighted sum of the set bit. Since bit 8 is the only bit used by system instrument, 256 is returned if the bit is set.

Bit 8 in the Condition register is cleared with the command:

DIAGNOSTIC:INTERRUPT:RESPONSE?

Unmasking the Operation Event Register Bits

When a condition monitored by the condition register occurs, a corresponding bit in the Operation Status Group Event register is automatically set. In order for this condition to generate a service request, the bit in the Event register must be unmasked using the Operation Status Group Enable register. This is done using the command:

STATUS:OPERATION:ENABLE <event>

where *event* is the decimal weight of the bit to be unmasked. Since the system instrument only uses bit 8, the only useful value of *event* is 256.

When bit 8 is set and is unmasked, it sets bit 7 in the Status Byte register in the Status Byte Group.

Bits in the Operation Status Group Event register which are unmasked can be determined with the command:

STATUS:OPERATION:ENABLE?

The command returns the decimal weighted sum of the unmasked bit(s).

Bits in the Operation Status Group Event register which are set can be determined with the command:

```
STATus:OPERation:EVENT?
```

This command returns the decimal weighted sum of the set bit(s).

Clearing the Operation Event Register Bits

Bits in the Operation Status Group Event register are cleared with the command:

```
STATus:OPERation:EVENT?
```

or the bits can be cleared with the command:

```
*CLS
```

The Operation Status Group Enable register is cleared (all bits masked) by sending the command:

```
STATus:OPERation:ENABle 0
```

Using the Operation Status Group Registers

The following example shows the sequence of commands used to setup and respond to an interrupt using the system instrument interrupt servicing routine.

NOTE

An interrupt handler must be assigned to handle the interrupt on the VXIbus backplane interrupt line specified. See "Interrupt Line Allocation" in Chapter 2 for more information.

```
!Call computer subprogram Intr_resp when a service request  
! is received due to an interrupt on a VXIbus backplane  
! interrupt line.
```

```
ON INTR 7 CALL Intr_resp
```

```
ENABLE INTR 7;2
```

```
!Unmask bit 7 in the Status Byte register so that a service  
! request (SRQ) will occur when an interrupt occurs.  
!Unmask bit 8 in the Operation Status Group Enable register  
!so that when the interrupt occurs it will set bit 7 in the  
!Status Byte register.
```

```
OUTPUT 70900; "**SRE 128"
```

```
OUTPUT 70900; "STAT:OPER:ENAB 256"
```

```
!Set up interrupt line 5 and enable interrupt response data  
!to be generated.
```

```
OUTPUT 70900; "DIAG:INT:SETUP5 ON"
```

```
OUTPUT 70900; "DIAG:INT:ACT ON"
```

```
• (Program which executes until interrupt occurs)
```

```
• !Computer service request routine which does an SPOLL  
!to determine the cause of the interrupt, then reads  
!(and clears) the Operation Event register to determine which  
!event occurred, and then reads the interrupt acknowledge  
! response (which also clears condition register bit 8).
```

```
SUB Intr_resp
  B = SPOLL(70900)
  OUTPUT 70900; "STAT:OPER:EVEN?"
  ENTER 70900; E
  OUTPUT 70900; "DIAG:INTR:RESP?"
  ENTER 70900; R
  .
  .
  .
SUBEND
```

Clearing Status

The *CLS command clears all status registers (Standard Event Status Register, Standard Operation Status Event Register, Questionable Data Status Event Register) and the error queue for an instrument. This clears the corresponding summary bits (bits 3, 5, & 7) and the instrument-specific bits (bits 0, 1, & 2) in the Status Byte Register. *CLS does not affect which bits are enabled to be reflected in the Status Byte Register or enabled to assert SRQ.

Interrupting an External Computer

When a bit in the status byte register is set and has been enabled to assert SRQ (*SRE command), the instrument sets the HP-IB SRQ line true. Interrupts can be used to alert an external computer to suspend its present operation and find out what service the instrument requires. (Refer to your computer/language manuals for information on how to program the computer to respond to the interrupt.)

To allow any of the status byte register bits to set the SRQ line true, you must first enable the bit(s) with the *SRE command. For example, suppose your application requires an interrupt whenever a message is available in the instrument's output queue (status byte register bit 4). The decimal weight of this bit is 16. You can enable bit 4 to assert SRQ by sending:

```
*SRE 16
```

NOTE

You can determine which bits are enabled in the Status Register using *SRE?. This command returns the decimal weighted sum of all enabled bits.

Example: Interrupting when an Error Occurs

This program shows how to interrupt an external computer whenever an error occurs for the instrument being programmed which, in this example, is a multimeter at secondary address 03.

```
10 OPTION BASE 1 !Array numbering starts with 1
20 ON INTR 7 CALL Errmsg !When SRQ occurs on interface 7, call subprogram
30 ENABLE INTR 7;2 !Enable SRQ interrupt, interface 7
40 OUTPUT 70903;"*SRE 32" !Enable bit 5 (Standard Event Status Bit) in Status Byte Register
50 OUTPUT 70903;"*ESE 60" !Enable error bits (bits 2-5) in Standard Event Status Register to be reflected in Status Byte Register
60 OUTPUT 70903;"MEAS:TEMP? TC,T,(@104)" !Measure temperature with voltmeter
70 WAIT 2
80 ENTER 70903;Tmp_rdg !Enter temperature reading
90 PRINT Tmp_rdg !Print temperature reading
100 END
110 SUB Errmsg
120 DIM Message$(256) !Create array for error message
130 CLEAR 70903 !Clear multimeter
140 B = SPOLL(70903) !Serial poll multimeter (clears SRQ)
150 REPEAT !Repeat next 3 lines until error number = 0
160 OUTPUT 70903;"SYST:ERR?" !Read error from queue
170 ENTER 70903;Code,Message$ !Enter error number & message
180 PRINT Code,Message$ !Print error number & message
190 UNTIL Code=0
200 OUTPUT 70903;"*CLS" !Clear status structures
210 STOP
220 SUBEND
```

Synchronizing an External Computer and Instruments

The *OPC? and *OPC commands (operation complete commands) allow you to maintain synchronization between an external computer and an instrument. The *OPC? query places an ASCII character 1 into the instrument's output queue when all pending instrument operations are finished. By requiring the computer to read this response before continuing program execution, you can ensure synchronization between one or more instruments and an external computer.

The *OPC command sets bit 0 (Operation Complete Message) in the Standard Event Status Register when all pending instrument operations are finished. By enabling this bit to be reflected in the Status Byte Register, you can ensure synchronization using the HP-IB serial poll function.

Example: Synchronizing an External Computer and Two Instruments using the OPC? query.

This example uses a D to A Converter module (DAC) at secondary address 09 and a Scanning Voltmeter at secondary address 03. The application requires the DAC to output a voltage to a device under test. After the voltage is applied, the voltmeter measures the response from the device under test. The *OPC? command ensures that the voltage measurement will be made only after the voltage is applied by the DAC.

```
10 OUTPUT 70909;"SOUR:VOLT1 5;*OPC?"  
    !Configure DAC to output 5 volts on channel 1; place 1 in  
    output  
    !queue when done  
20 ENTER 70909;A  
    !Wait for *OPC? response  
30 OUTPUT 70903;"MEAS:VOLT:DC? (@104)"  
    !Measure DC voltage on device under test  
40 ENTER 70903;A  
    !Enter voltage reading  
50 PRINT A  
    !Print reading  
60 END
```

Example: Synchronizing an External Computer and Two Instruments using the *OPC command.

This example uses the *OPC command and serial poll to synchronize an external computer and two instruments (DAC at secondary address 09; Scanning Voltmeter at secondary address 03). The advantage to using this method over *OPC? query method is that the computer can do other operations while it is waiting for the instrument(s) to complete operations. When using this method, the Operation Complete bit (bit 0) must be the only enabled bit in the Standard Event Status Register (*ESE 1 command). If other bits (such as error bits) are enabled, you must make sure that bit 0 causes the interrupt.

```
10 OUTPUT 70909;"*CLS"  
    !Clear all status structures on instrument at secondary address  
    09  
20 OUTPUT 70909;"*ESE 1"  
    !Enable Operation Complete to be reflected in bit 5 of the  
    Status Byte Register  
30 OUTPUT 70909;"SOUR:VOLT1 5;*OPC"  
    !Configure instrument #1, set Operation Complete bit when  
    done  
40 WHILE NOT BIT(SPOLL(70909),5)  
    !While waiting for bit 5 in instrument's Status Byte Register to  
    be set,  
    !computer can do other operations  
50 !(Computer does other operations here)  
60 END WHILE  
70 OUTPUT 70903;"MEAS:VOLT:DC? (@104)"  
    !Measure DC voltage using instrument #2  
80 END
```


Downloading Device Drivers

About this Chapter

This chapter describes the procedure for using downloadable device drivers with the E1405 Command Module. This functionality was added so that SCPI capability for new register based devices could be added to the Command Module without having to update an internal set of ROMs. This chapter contains the following sections:

- About this Chapter 5-1
- What You Will Need 5-1
- Memory Configuration 5-3
- Download Program Configuration 5-4
- Downloading Drivers in MS-DOS systems 5-6
- Downloading Drivers in HP IBASIC Systems 5-7
- Downloading Drivers from Other HP BASIC Systems 5-8
- Downloading Multiple Drivers 5-9
- Checking Driver Status 5-9
- Manually Downloading Drivers 5-10

What You Will Need

The downloadable device drivers and the software necessary to download the drivers into HP mainframes are provided on 3.5" floppy disks which ship with the device driver manual. Disks are provided in both LIF and DOS format for your convenience. Drivers and appropriate downloading software are provided for use in MS-DOS systems downloading over an RS-232 link and for use in systems using HP BASIC or HP IBASIC (Instrument BASIC) and downloading over an HP-IB (IEEE 488.2) link. The procedures for both types of downloaders are detailed later in this chapter.

Figure 5-1 shows the files and documents that will be needed for each type of download supported.

For RS-232 downloads you will need appropriate cables to connect your computer to the Command Module. If your computer has a 25 pin serial output connector, you can use an HP 24542G cable to make the connection. If your computer has a 9 pin serial output connector, you can use an HP 24542M *and* an HP 24542H cable (connected end to end) to make the connection.

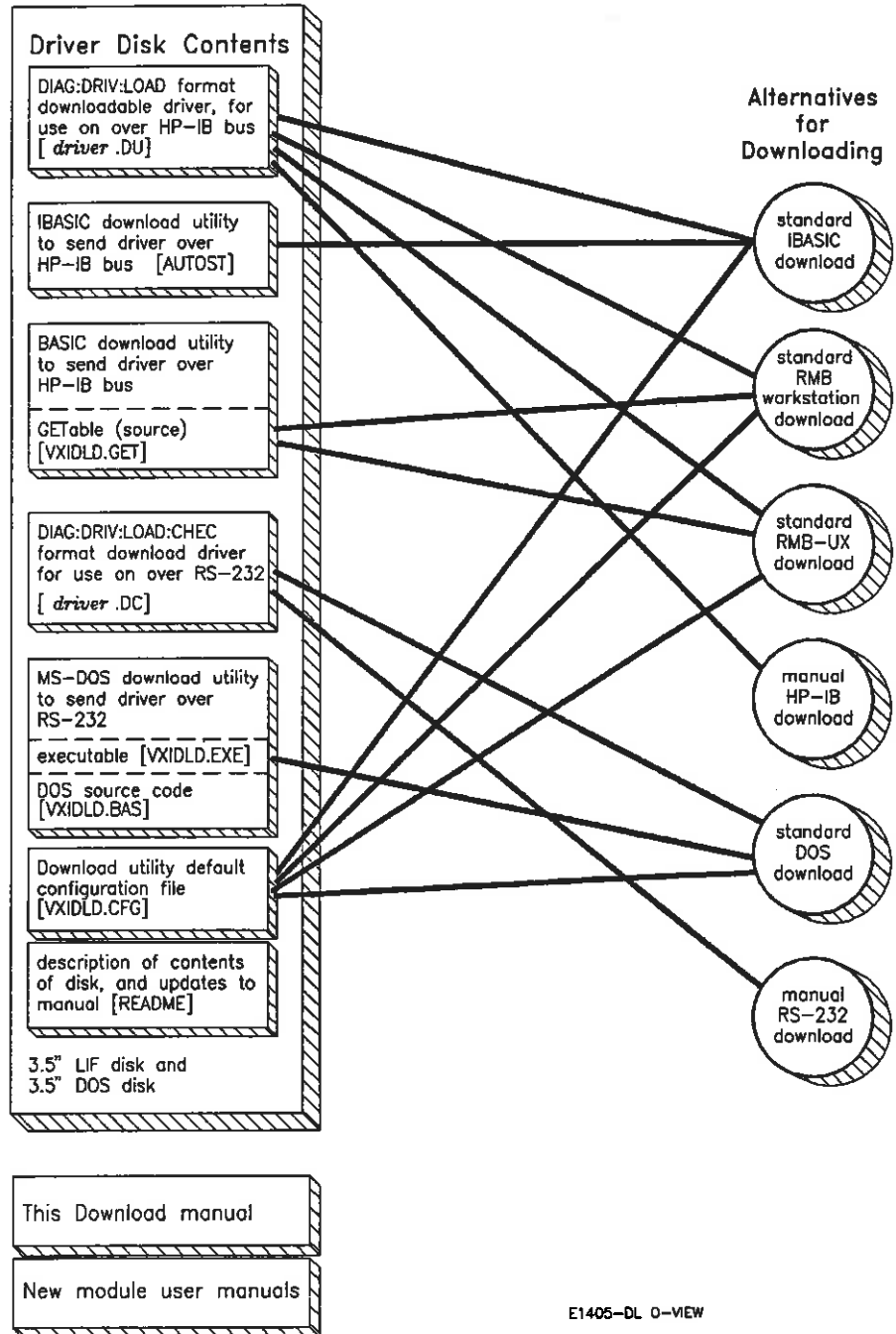


Figure 5-1. Driver and Documentation Usage

Memory Configuration

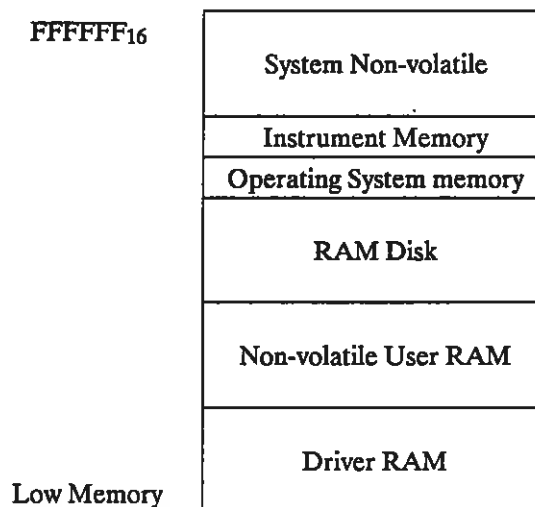
Before attempting to download any device drivers you should understand how memory is affected when you specify a size for one or more types of RAM. There are three types of RAM that you can allocate in the mainframe:

- RAM disk (RDISK)
- Non-volatile RAM (NRAM)
- Driver RAM (DRAM)

Figure 5-2 shows the positioning of these areas in memory. User Non-volatile RAM and RAM Disk both occupy higher memory addresses than the Driver RAM. Because the actual size of these three areas is variable, they do not have a fixed starting position. At creation time, the lowest unused memory address becomes the starting address for the requested type of RAM. Memory areas set at higher addresses can be created without affecting any previously created lower memory areas, but creating a new memory area causes any areas *above it* to be removed.

NOTE

If you wish to use RDISK or NRAM, you can modify the configuration file so that the download program sets up the required memory segments.



The Low Address depends on the amount of memory installed. It is equal to the highest address plus 1 (1000000_{16}) minus the size of memory installed. The boot time messages will tell you how much RAM you have installed in your system. In an E1405 with 512Kbytes of memory the Low Address is
 $low\ address = 1000000_{16} - 80000_{16} = F80000_{16}$, or 16,252,928 decimal.

Figure 5-2. Positioning of Allocatable RAM

Example *If you create a RAM Disk area without creating any User Non-volatile RAM or Driver RAM, the starting address for the RAM Disk will be at the lowest address ($F80000_{16}$ for a command module with 512Kbytes of memory). If you now create a Driver RAM area, the RAM Disk area will be removed since the new area has to be at a lower address than the RAM Disk area.*

Download Program Configuration

If you will not be using the default configurations for downloading, you will need to edit the configuration file to match your system configuration. If the default values shown below are correct for your setup, you can proceed to the appropriate downloading instructions.

The configuration defaults for MS-DOS systems are:

- Download program searches for drivers in current directory.
- Execution Log is OFF (log to screen only).
- All drivers in current directory will be downloaded.
- COM1 is used for output.
- Baud rate is 9600.
- 1 stop bit is used.
- NRAM size is zero.
- RDISK size is zero.

The configuration defaults for HP-IB systems are:

- Download program searches for drivers in current directory.
- Execution Log is OFF (log to screen only).
- All drivers in current directory will be downloaded.
- 80900 is used for the interface address when running from IBASIC. 70900 is used as the interface address when running in any HP BASIC environment other than IBASIC.
- NRAM size is zero.
- RDISK size is zero.

Editing the Configuration File

The configuration file (VXIDLD.CFG or VXIDLD_CFG) on your driver distribution disk is shipped with all entries commented out. In this state, the download programs will use the default values shown above. To activate or change an entry, you must edit the file manually. The file is set up so that it can be edited either by a standard text editor or word processor, or with a Basic language editor. Comments and instructions are included in the file.

- The beginning of the useful information on each line is the part following "*linenumber* REM" (the "*linenumber* REM" is ignored).
- All lines beginning with "#" are comments.
- Lines that start with "##" are intended to remain comments.
- Lines that start with "# " are example lines that you may wish to activate and/or modify. These are the actual configuration statements.
- Setting labels are not case sensitive, and should be separated from the associated value by an equal sign ("=").
- Unrecognized settings are ignored.
- If you activate more than one line for a setting that can take only one value, the first value found for the setting will be used.

DIRECTORY= specifies the directory where you store your drivers and where the driver programs will log information about their progress. The default is the current directory. The directory specified must be writeable if you are doing downloads using IBASIC or logging progress.

EXECUTION LOG = specifies the place to log information about the program's progress. The default location for this function is the screen. If you

specify a file name here, the driver downloader will log to the screen and to the specified file.

DRIVER FILE = specifies the driver file or files to download. The default is to download all device driver files found in the directory specified by **DIRECTORY** = . If the driver downloader finds one line in this format, it will assume that you are specifying entries and will only download the listed entries. This configuration item can have multiple lines.

ADDRESS = specifies the I/O interface that you will be using. The default interface address when running in IBASIC over HP-IB is 80900. The default address when running over HP-IB in any other HP BASIC environment is 70900. The default address when running in DOS is 1 (for COM1:).

The communication interface you will be using when running from any of Hewlett-Packard's BASIC environments is the "HP-IB" interface (also known as IEEE 488.1). Selection of a specific HP-IB interface consists of an address in the form "sspp00" where:

ss is the select code of the HP-IB interface card.

pp is the primary HP-IB address used for the VXI mainframe.

00 is the secondary HP-IB address used for the SYSTEM instrument.

The communication interface you will be using when running from DOS is the "RS-232" interface. When Using the RS-232 interface the serial cable must be connected to either the built-in RS-232 connection of the VXI mainframe or an RS-232 module (HP E1324A) that is set to interrupt at the default interrupt level (level 1). Selection of the address for the RS-232 interface consists of an address that is 1 for COM1 or 2 for COM2:.

BAUD = specifies the baud rate of the transmission if you are using RS-232. The default is 9600 (which is also the default for the VXI mainframe after a DIAG:BOOT:COLD command). Allowed values are 300, 1200, 2400, 4800, 7200, or 9600 (19,200 is not supported by DOS).

STOP BITS = specifies the number of stop bits per byte if you are using RS-232. The default is 1 (which is also the default for the VXI mainframe after a DIAG:BOOT:COLD command). Allowed values are 1 or 2.

NRAM = specifies the size in bytes of the non-volatile user RAM area you wish to set up. The default value is zero bytes. You may change this value later independent of the downloaded drivers, but changing it will always affect any RAM disk (RDISK) you have specified.

RDISK = specifies the size in bytes of the RAM disk segment you wish to set up. The default value is zero bytes. You can change this value later without affecting either the downloaded device drivers or the user non-volatile RAM (NRAM).

Downloading Drivers in MS-DOS Systems

The device driver download program VXIDLD.EXE provided on the disk with the driver files for use with an RS-232 interface must be run from MS-DOS. It will set up the the required device driver memory and any other memory partitions defined in the configuration file, reboot the system, and download the device driver. If there are device drivers present, or you already have memory allocated for NRAM (User Non-volatile RAM) or RDISK (RAM Disk), a warning will be issued and the downloading process aborted. You must first clear any existing drivers from the system, and then download all of the required drivers together. You may redefine any NRAM or RDISK areas after downloading the device drivers.

1. Make sure that your computer can talk to the E1405 Command Module. If you have changed the communications protocol for the Command Module or mainframe, you must change them back to 9600 BAUD, 8 data bits, 1 stop bit, and no parity before this download will work correctly.

These are the defaults after cold boot. If necessary, you can change the baud rate and number of stop bits in the configuration file, but since the special formatting required for downloading over RS-232 requires all 8 data bits in each byte, you must make sure that the data bits are set to 8 and parity checking is OFF. The download program handles its own pacing, so the setting for pacing does not matter.

2. Put the floppy disk into an appropriate drive.
3. Make sure that the floppy disk is your current drive (for example, type "A:" and press ENTER).
4. Execute the device downloader program (type "VXIDLD" and press ENTER).
5. The downloader program will check to make sure that there are no device drivers already loaded, and no memory has been allocated for NRAM or RDISK. If either condition exists, the program will issue a warning and abort. If not, it will create the required RAM partitions, reboot the system, and download the device driver on the supplied disk.

Any errors encountered while downloading will be reported.

6. The download program will check to make sure that the driver has been downloaded and is in memory.

WARNING

Terminate and Stay Resident programs in your MS-DOS system may interfere with the timing of RS-232 transfers and cause errors in the downloading. If you encounter errors indicating that the download program did not receive back what it expected, and the driver is not loaded, remove all of your TSRs from memory and try the download procedure again.

Downloading Drivers in HP-IB Systems with IBASIC

The device driver download program AUTOST provided on the disk with the driver files for use with HP-IB must be run from HP IBASIC (Instrument Basic). It will set up the the required device driver memory and any other memory partitions defined in the configuration file, reboot the system, and download the device driver. This program will issue a warning and abort if any errors are encountered. If there are device drivers present, or if you already have memory allocated for NRAM (User Non-volatile RAM) or RDISK (RAM Disk), you must first clear any existing drivers from the system, and then download all of the required drivers together. You may redefine any NRAM or RDISK areas after downloading the device drivers.

NOTE

If you wish to see the messages that the download program generates, you need to have a terminal connected to the IBASIC display port. If you have not changed this from its default value of NONE, messages are sent to the built-in RS-232 port.

-
1. Make sure that your Command Module (E1405) is set to System Controller mode.
 2. Put the floppy disk into an appropriate drive.
 3. Make sure that the floppy disk is your current drive (for example, type 'MSI ";,700,1" and press ENTER).
 4. Load the device download program into IBASIC (type 'GET "AUTOST"' and press ENTER) and run the program (type "RUN" and press ENTER).
 5. The download program will check to make sure that there are no device drivers already loaded, and no memory has been allocated for NRAM or RDISK. If either condition exists, the program will issue a warning and abort. If not, it will create the required RAM partitions, reboot the system, and download the device driver on the supplied disk.

Any errors encountered while downloading will be reported and will cause the program to abort.
 6. The download program will check to make sure that the driver has been downloaded and is in memory.

NOTE

If you are using IBASIC but controlling the system over the HP-IB, you must put all commands in quotes and prefix them with "PROG:EXEC". A typical command would be:

```
PROG:EXEC 'MSI ";,700,1"
```

Downloading Drivers in HP-IB Systems with HP BASIC

The device driver download program `VXIDLD_GET` provided on the disk with the driver files for use with HP-IB must be run from an HP BASIC other than IBASIC. It will set up the the required device driver memory and any other memory partitions defined in the configuration file, reboot the system, and download the device driver. If there are device drivers present, or you already have memory allocated for NRAM (User Non-volatile RAM) or RDISK (RAM Disk), a warning will be issued and the downloading process aborted. You must first clear any existing drivers from the system, and then download all of the required drivers together. You may redefine any NRAM or RDISK areas after downloading the device drivers.

1. Make sure that your Command Module (E1405) *is not* set to System Controller mode.
2. Put the floppy disk into an appropriate drive.
3. Make sure that the floppy disk is your current drive (for example, type `'MSI ";,700,1"` and press ENTER).
4. Load the device download program into BASIC (type `'GET "VXIDLD_GET"` and press ENTER) and run the program (type `"RUN"` and press ENTER).
5. The download program will check to make sure that there are no device drivers already loaded, and no memory has been allocated for NRAM or RDISK. If not, it will create the required RAM partitions, reboot the system, and download the device driver on the supplied disk.

Any errors encountered while downloading will be reported and will cause the program to abort.

6. The download program will check to make sure that the device driver was successfully downloaded.

Downloading Multiple Drivers

The driver downloader software automatically checks for the existence of other drivers when it is run. If there are device drivers present, it will abort the process and inform you that you must first clear the other device drivers out of the mainframe and then download all of the required drivers at once. The easiest way to accomplish this is to place copies of all of the device drivers into a single directory on your hard disk along with the downloader, or onto the same floppy disk. The download program will look in its own directory first, and download any device drivers it finds.

1. Move all of your device drivers into a single directory with the downloaders.
2. Clear the DRAM memory in the mainframe (send "DIAG:DRAM:CRE 0" and "DIAG:BOOT" to the System Instrument).
3. Execute or load and run the appropriate device driver software, as described above.

All device drivers in the directory or on the same floppy disk as the driver downloader will be downloaded automatically after the system checks to make sure that there are no other device drivers already loaded. You can change several aspects of the downloading procedure by editing the configuration file .

Checking Driver Status

Once your drivers are downloaded, you can use the System Instrument command `DIAG:DRIV:LIST?` to check their status. In the format shown, this command lists all types of drivers. You can specify the *type* (ALL, RAM or ROM) by using `DIAG:DRIV:LIST:type?`

- `DIAG:DRIV:LIST?` lists all drivers in the system.
- `DIAG:DRIV:LIST:RAM?` lists all drivers found in the RAM driver table DRAM. These are the drivers which you just downloaded into the system.
- `DIAG:DRIV:LIST:ROM?` lists all drivers found in the ROM driver table. These drivers are always present in the system. If one of these is meant for an instrument which also has a driver in RAM, the driver in RAM will be used by the system.

Manually Downloading a Driver

Download programs are supplied for use with the system setups described earlier in this chapter. If you have a system setup that does not allow the use of one of the supplied download programs (for instance, if you are using a Macintosh® computer), you will need to manually download the driver. The details of this process will be different for different system setups, but the basic procedures are outlined below.

Preparing Memory for Manual Downloading

Before you can manually download any drivers using either RS-232 or HP-IB, you must define the DRAM (Downloadable RAM) into which the drivers will be transferred. DRAM memory is non-volatile.

1. Calculate the required total DRAM size. This is the total amount of memory required by the mainframe for all of the device drivers you are going to download.

Typical driver size will range from 40Kbytes to 100Kbytes. If you are in doubt about the amount of memory needed for downloading your device drivers, use the size of the HP-IB driver file (ends in "DU") on the driver disks. Remember that you must add the amount of memory necessary for all of the device drivers you plan to download. You can see how much RAM is available by using the DIAG:DRAM:CRE? MAX,DEF query.

NOTE

Each driver will need additional system RAM at run time. Although this is not part of the RAM necessary for the DRAM calculations, you should make sure that you have enough DRAM to download the drivers, and enough system RAM left after downloading to run the drivers. Most drivers will need less than 15Kbytes of additional RAM (per driver) at run time. If IBASIC is in the system, it will take at least 150Kbytes to 200Kbytes of system RAM in addition to the RAM used by the device drivers.

-
2. Create the appropriate DRAM partition using the DIAG:DRAM:CRE command. Unless you have more than eight drivers to download, you do not need to specify the second parameter.

WARNING

Creating this memory partition will delete any NRAM or RDISK partitions that you have defined, and any data in NRAM or RDISK memory. You must redefine any such memory blocks after you have defined the Driver RAM.

-
3. Reboot the system

Manually Downloading Over HP-IB

Manually downloading a driver over HP-IB is fairly straightforward. This discussion assumes that the downloadable device driver has been supplied by Hewlett-Packard. Drivers supplied by HP are formatted so that you just need to transfer the driver to command module memory. You must also have the driver on media that is accessible to the host computer that will be controlling the download.

You should send a *RST command and a *CLS command to the SYSTEM instrument to put it in a known state before beginning your download.

On most computers, a program will be required for the actual download process. Since the driver file contains the System Instrument command to start the downloading and the actual data to download, this program just needs to transfer the bytes in the driver file to the System Instrument, one byte at a time.

This file contains the SCPI command DIAG:DRIV:LOAD followed by the IEEE 488.2 arbitrary definite block header, and then the actual driver. The definite block starts with the # character, followed by a single digit that shows how many digits are in the length field, followed in turn by the length field. For instance, a block that is 1000 bytes long would have a block header of #800001000.

When your transfer program is complete you should send the SCPI query SYST:ERR? to make sure that there were no errors during the download, and reboot the system (send DIAG:BOOT). You can make sure that all of your drivers have been properly loaded into Driver RAM by sending the SCPI command DIAG:DRIV:LIST:RAM?

Manually Downloading Over RS-232

Manually downloading a driver over RS-232 is similar in concept to downloading over HP-IB. Drivers supplied by HP are formatted so that you just need to transfer them to command module memory. You must also have the driver on media that is accessible to the host computer that will be controlling the download.

However, the RS-232 interface of the E1405 uses special control characters (e.g., <CTRL-C> to implement the equivalent of the HP-IB "device clear" function) that would cause havoc in the download process if sent as part of the driver. The driver file on the distribution disk that ends in "DC" is specially formatted for RS-232 downloading to avoid this problem (see Appendix D "Formatting Binary Data for RS-232" for more information on the data format of these files).

Transmission Format

You need to make sure that the transmission format of your computer matches the format used at the System Instrument. The default configuration for the System Instrument after a DIAG:BOOT:COLD command has been issued is

- 9600 BAUD
- 8 data bits
- 1 stop bit
- Parity checking is OFF
- XON/XOFF pacing

If you are going to use any other setting, you must set up the appropriate settings in the System Instrument using the following commands

COMM:SER[n]:REC:BAUD <rate>	<i>sets BAUD rate</i>
COMM:SER[n]:REC:SBITS <bits>	<i>sets number of stop bits</i>
DIAG:COMM STOR	<i>saves settings so they will be kept through a reboot.</i>

NOTE

Because the special formatting for binary files uses all 8 bits, the number of data bits must be set to 8 and parity checking must remain OFF for the driver files to transfer properly.

Pacing the Data

Since the RS-232 interface is asynchronous, it is possible for the computer that is doing the download to overrun the System Instrument. This would cause part of the driver to be lost. To prevent this from happening, you should enable hardware handshake (either RTS or DTR) or software handshake (XON/XOFF).

The default configuration for the E1405 Command Module is for software handshake enabled and hardware handshake disabled. To make sure that software handshake is enabled for the command module use the SYST:COMM:SER:PACE? query. To set up software handshake you can use the following commands:

```
SYST:COMM:SER:PACE:THR:STOP? MAX
    to find the maximum number of characters to fill the input
    buffer.
SYST:COMM:SER:PACE:THR:STOP <max-20>
    to set the threshold for stopping data to the maximum size of
    the input buffer minus 20 characters.
SYST:COMM:SER:PACE:THR:STAR 0
    to set the start buffer level to zero. This makes sure that the
    input buffer is completely flushed whenever transmissions are
    stopped.
SYST:COMM:SER:PACE:XON
    to enable the software handshake protocol.
```

The start threshold is not critical as long as it is less than the stop threshold. The stop threshold must be set low enough to handle the maximum number of characters that are likely to be received at the System Instrument after it sends the XOFF signal.

Hardware handshake can be set up to use either the DTR (Data Terminal Ready) line or the RTS (Ready to Send) line. These modes can be set with the SYST:COMM:SER:CONT:DTR IBFULL command (to set for DTR) or SYST:COMM:SER:CONT:RTS IBFULL command (to set for RTS). You should also make sure that software handshake is OFF by using the SYST:COMM:SER:PACE NONE command. When the input buffer of the System Instrument is not full (number of characters in the input buffer is less than the high threshold), the specified hardware line will be asserted. When either hardware handshake mode is enabled, the System Instrument will not

transmit characters when either the CTS (Clear to Send) or the DSR (Data Set Ready) lines are not asserted. This acts to pace the System Instrument output.

NOTE

The E1405 Command Module RS-232 interface is implemented as a DTE (Data Terminating Equipment). Since most computer RS-232 interfaces are also implemented as DTEs, a cable that does line swapping (null modem cable) is usually used to connect the computer to the instrument. This cable typically swaps the receive and transmit lines. It will usually connect the DTR line of one interface to the CTS and DSR lines of the other. It will connect the RTS line of one interface to the DCD (Data Carrier Detect) line of the other.

CAUTION

The RS-232 interface of the E1405 Command Module will echo any characters received with an ASCII value greater than 32 and less than 128. Carriage returns are echoed as carriage return/linefeed. When transferring the driver file, these echoes can fill up the RS-232 receive buffer of your computer if they are not read. If receive pacing is enabled for your computer this could cause the computer to send the "Stop Transmitting" signal to the System Instrument, which could block the remaining downloaded bytes or other commands sent after the download. Since the driver file contains command strings and many carriage returns that will be echoed by the system, your program should read the returning echo characters from the RS-232 line. This will also let you determine if there are any error messages coming back.

Transmitting Using a COPY Command

On some computers it is possible to use an RS-232 or HP-IB port and the copy command to transfer the device driver. Hardware or software handshake must be used by the copy command on the computer doing the downloading, and the same handshake mode must be enabled on the System Instrument.

1. Set the required handshake mode and data format (e.g., on DOS systems use the MODE command).
2. Type "COPY *filename port*" to transfer the file through the RS-232 port to the System Instrument (e.g., on a DOS system you might use "COPY /B *filename*.DC COM1:"). This command may be slightly different depending on the type of computer being used.

NOTE

Since errors are echoed immediately, this method of transfer has no means of trapping errors.

Transmitting Using a CAT Command

On HP-UX systems you can use the *cat* command to transfer the device driver. The appropriate device file must exist. All shell commands are assumed to be executed from either the */bin/sh* or */bin/ksh* shell.

1. Start a process that opens the device file to be used. This process should keep the device file open long enough for the transfer to begin. This step is done so that the following command to set the device file configurations will remain in effect for the transfer. A command that will do this is:

```
(cat < device file > /dev/null; sleep 1000) &
```

2. Set the required configuration of the device file using the *stty* command. The following command will set the device file to work with the default System Instrument configuration.

```
stty -opost 9600 ixon -ixoff cs8 -cstopb ignpar < device file
```

3. Transfer the file to the System instrument with the *cat* command.

```
cat filename > device file
```

Transmitting Using Custom Software

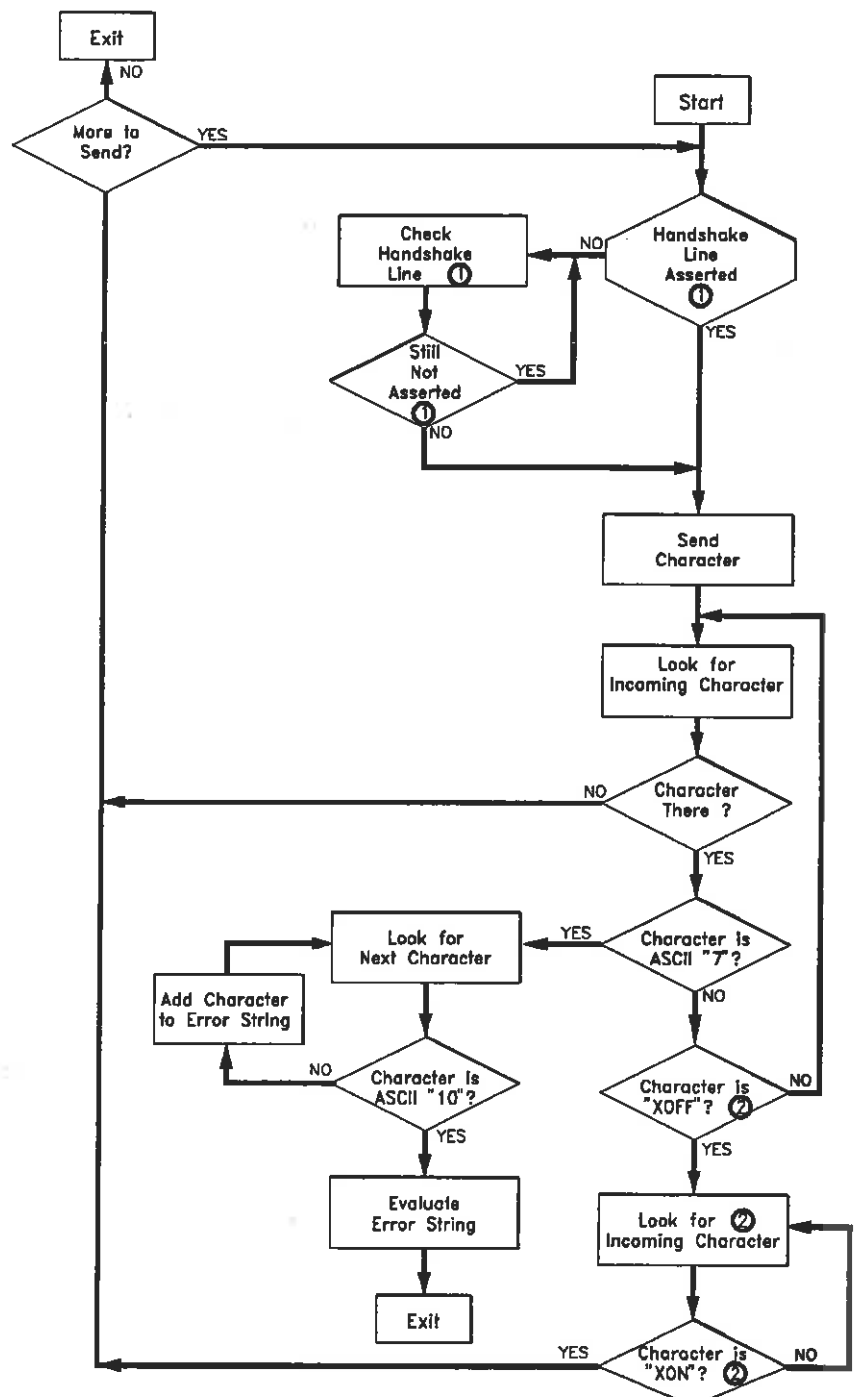
If the COPY command on your computer cannot directly implement handshaking, or if you wish to trap errors and abort or otherwise modify the transmission process, you must use a program to handle the download process.

This procedure assumes that your computer has some means of looking at data being echoed from the System Instrument, and can check for a return character without having to have a character returned. Since the actual driver file bytes sent over the RS-232 interface are not echoed, the lack of ability to do this would put the system into an infinite wait at the first byte that was not echoed.

1. Set up the appropriate handshake mode and data format on your system, and the matching handshake mode in the System Instrument.
2. Transfer the driver file over the RS-232 interface using a program that follows the outline in figure 5-3.

Check Driver Status

Make sure that the drivers were properly downloaded by checking their status using the *DIAG:DRIV:LIST:RAM?* command. This will give you a list of all the drivers currently found in DRAM.



- ① Skip these steps if using software handshake.
- ② Skip these steps if using hardware handshake.

E1405-DL FIG5-3

Figure 5-3. Manually Downloading a Device Driver

E1405 Command Reference

About This Chapter

This chapter describes the **Standard Commands for Programmable Instruments (SCPI)** command set and the **IEEE 488.2 Common Commands** for the System Instrument. The System Instrument is part of the HP E1405 Mainframe's internal control processor and is therefore always present in a Mainframe. This chapter contains the following sections:

- **Command Types** 6-1
- **SCPI Command Reference** 6-4
- **Common Command Reference** 6-83
- **HP-IB Message Reference** 6-90
- **Command Quick Reference** 6-93

Command Types

Commands are separated into two types: **IEEE 488.2 Common Commands** and **SCPI Commands**.

Common Command Format

The IEEE 488.2 standard defines the **Common commands** that perform functions like reset, self-test, status byte query, etc. Common commands are four or five characters in length, always begin with the asterisk character (*), and may include one or more parameters. The command keyword is separated from the first parameter by a space character. Some examples of Common commands are shown below:

```
*RST, *ESE <mask>, *STB?
```

SCPI Command Format

The SCPI commands perform functions like closing switches, making measurements, and querying instrument states or retrieving data. A subsystem command structure is a hierarchical structure that usually consists of a top level (or root) command, one or more lower level commands, and their parameters. The following example shows part of a typical subsystem:

```
[ROUTE:]
  CLOSe <channel_list>
  SCAN <channel_list>
  :MODE?
```

ROUTE: is the root command, CLOSe and SCAN are second level commands with parameters, and :MODE? is a third level command.

Command Separator A colon (:) always separates one command from the next lower level command as shown below:

ROUTe:SCAN:MODE?

Colons separate the root command from the second level command (ROUTe:SCAN) and the second level from the third level (SCAN:MODE?).

Abbreviated Commands The command syntax shows most commands as a mixture of upper and lower case letters. The upper case letters indicate the abbreviated spelling for the command. For shorter program lines, send the abbreviated form. For better program readability, you may send the entire command. The instrument will accept either the abbreviated form or the entire command.

For example, if the command syntax shows MEASure, then MEAS and MEASURE are both acceptable forms. Other forms of MEASure, such as MEASU or MEASUR will generate an error. You may use upper or lower case letters. Therefore, MEASURE, measure, and MeAsUrE are all acceptable.

Implied Commands Implied commands appear in square brackets ([]) in the command syntax. (The brackets are not part of the command, and are not sent to the instrument.) Suppose you send a second level command but do not send the preceding implied command. In this case, the instrument assumes you intend to use the implied command and it responds as if you had sent it. Examine the SOURCE subsystem shown below:

```
[SOURCE:]
  PULSe
    :COUNT
    :COUNT?
    :PERiod
    :PERiod?
```

The root command SOURCE: is an implied command. To set the instrument's pulse count to 25, you can send either of the following command statements:

SOUR:PULS:COUN 25 *or* PULS:COUN 25

Variable Command Syntax Some commands have what appears to be a variable syntax. For example:

DIAG:INT:SETup[n]? and SYST:COMM:SERial[n]:BAUD?

In these commands, the "n" is replaced by a number. No space is left between the command and the number because the number is not a parameter. The number is part of the command syntax. The purpose of this notation is to save a great deal of space in the command reference. In the case of ...SETup[n], n could range from 1 through 7. In ...SERial[n]..., n can be from 0 through 7. You can send the command without the [n] and a default value will be used by the instrument. Some examples:

DIAG:INT:SETUP2?, DIAG:INT:PRI2 5, SYST:COMM:SER1:BAUD 9600

Parameters **Parameter Types.** The following list contains explanations and examples of parameter types you will see later in this chapter.

- **Numeric Parameters** are commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation

(e.g., 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01). Special cases include MIN, MAX, and INFINITY. The Comments section within the Command Reference will state whether a numeric parameter can also be specified in hex, octal, and/or binary. #H7B, #Q173, #B1111011

- **Boolean parameters** represent a single binary condition that is either true or false (e.g., ON, OFF, 1, 0). Any non-zero value is considered true.

Discrete parameters select from a finite number of values. These parameters use mnemonics to represent each valid setting. An example is the TRIGGER:SOURCE <source> command where *source* can be BUS, EXT, HOLD, or IMM.

- **Arbitrary Block Program Data parameters** are used to transfer blocks of data in the form of bytes. The block of data bytes is preceded by a preamble which indicates either 1) the number of data bytes which follow, or 2) that the following data block will be terminated upon receipt of a New Line message with the EOI signal true. The syntax is:

Definite Length Block

< non-zero digit > < digit(s) > < data byte(s) >

Where the value of < non-zero digit > equals the number of < digit(s) >. The value of < digit(s) > taken as a decimal integer indicates the number of < data byte(s) > in the block.

Indefinite Length Block

#0 < data byte(s) > < NL ^ END >

Examples of sending 4 data bytes:

```
#14 < byte > < byte > < byte > < byte >
#3004 < byte > < byte > < byte > < byte >
#0 < byte > < byte > < byte > < byte > < NL ^ END >
```

Optional Parameters. Parameters shown within square brackets ([]) are optional parameters. (Note that the brackets are not part of the command, and are not sent to the instrument.) If you do not specify a value for an optional parameter, the instrument chooses a default value. For example, consider the ARM:COUNT? [< MIN | MAX >] command. If you send the command without specifying a parameter, the present ARM:COUNT value is returned. If you send the MIN parameter, the command returns the minimum count available. If you send the MAX parameter, the command returns the maximum count available. Be sure to place a space between the command and the parameter.

Linking Commands

Linking IEEE 488.2 Common Commands with SCPI Commands. Use a semicolon between the commands. For example:

```
*RST;OUTP ON or TRIG:SOUR HOLD;*TRG
```

Linking Multiple SCPI commands. Use both a semicolon and a colon between the commands. For example:

```
ARM:COUN 1;;TRIG:SOUR EXT
```

SCPI Command Reference

This section describes the SCPI commands for the System Instrument. Commands are listed alphabetically by subsystem and also within each subsystem. A command guide is printed in the top margin of each page. The guide indicates the first command listed on that page.

DIAGnostic

The DIAGnostic subsystem allows control over the System Instrument's internal processor system (:BOOT, and :INTerrupt), the allocation and contents of User RAM, and, disc volume RAM (:NRAM, and :RDISk), and allocation of the built-in serial interface (:COMM:SER:OWNer).

Subsystem Syntax

```

DIAGnostic
:BOOT
  :COLD
  [:WARM]
:COMMunicate
  :SERial[0]
    [:OWNer] [SYSTem|IBASi c|NONE]
    [:OWNer]?
  :SERial[n]
  :STORe
:DOWNload
  :CHECked
    [:MADDress] <address>, <data>
    :SADDress <address>, <data>
    [:MADDress] <address>, <data>
    :SADDress <address>, <data>
:DRAM
  :AVAlable?
  :CREate <size>, <numdrivers>
  :CREate? [<MIN|MAX>, <MIN|MAX|DEF> ]
:DRIVer
  :LOAD <driver_block>
  :CHECked <driver_block>
  :LIST
    :ALL?
    :RAM?
    :ROM?
:INTerrupt
  :ACTivate [0|1|ON|OFF]
  :SETup[n] [0|1|ON|OFF]
  :SETUP[n]?
  :PRiority[n] [<priority> |MIN|MAX|DEF]
  :PRiority[n]? [MIN|MAX|DEF]
  :RESPonse?
:NRAM
  :ADDRess?
  :CREate <size> |MIN|MAX
  :CREate? [MIN|MAX]
:PEEK? <address>, <width>
:POKE <address>, <width>, <data>
:RDISk
  :ADDRess?
  :CREate <size> |MIN|MAX
  :CREate? [MIN|MAX]
:UPLoad
  [:MADDress]? <address>, <byte_count>
  SADDress? <address>, <byte_count>

```

:BOOT:COLD **DIAGnostic:BOOT:COLD** causes the System Instrument to restart (re-boot). Configurations stored in non-volatile memory and RS-232 configurations are reset to their default states:

- DRAM, NRAM, and RDISk memory segments are cleared.
- Serial Interface parameters for the internal serial interface and for any plug-in serial cards (E1324) that are in the Command Module's servant area are set to:
 - BAUD 9600
 - BITS 8
 - PARity NONE
 - SBITs 1
 - DTR ON
 - RTS ON
 - PACE XON
- Serial 0 Owner = system

NOTE

Resetting the serial interface parameters takes about 0.01 seconds for the built-in serial port and 0.75 seconds per serial plug-in card. While this is taking place the System Instrument will still respond to serial polls. If you are using a serial poll to determine when the cold boot cycle is complete, you should insert a delay of 1 second per plug-in serial card (E1324) before polling the system instrument. This will prevent incorrectly determining that the system instrument has completed its boot cycle.

- Comments**
- The System Instrument goes through its power-up self tests.
 - **Related Commands:** DIAG:BOOT:WARM

Example Re-booting the System Instrument (cold)

DIAG:BOOT:COLD *force boot*

:BOOT[:WARM] **DIAGnostic:BOOT[:WARM]** causes the System Instrument to restart (re-boot) using the current configuration stored in non-volatile memory. The effect is the same as cycling power.

- Comments**
- The System Instrument goes through its power-up self tests.
 - The non-volatile system state is used for configuration wherever applicable.
 - **Related Commands:** DIAG:BOOT:COLD

Example Booting the System Instrument (warm)

DIAG:BOOT:WARM *force boot*

**:COMMunicate
:SERial[0][:OWNer]**

DIAGnostic:COMMunicate:SERial[0][:OWNer] <owner > Allocates the built-in serial interface to the System Instrument, the optional IBASIC interpreter, or to neither.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
owner	discrete	SYSTEM IBASic NONE	none

Comments

- While the serial interface is allocated to the Command Module (SYSTEM), it can function as the mainframe user interface when connected to a terminal or computer running terminal emulation software.
- When the built-in serial interface is allocated to IBASIC, it is controlled only by IBASIC. The serial interface is given a select code of 9, and any RS-232 device connected to the (Command Module) RS-232 port is programmed accordingly. Note that when IBASIC owns the serial interface there is no "front panel" interface to the system.
- If the built-in serial interface is not needed, specifying NONE will release memory for use by other instruments.
- Once the new serial interface owner has been specified (DIAG:COMM:SER:OWN), the change will not take effect until you re-boot (warm) the system.
- **Related Commands:** DIAGnostic:COMMunicate:SERial[:OWNer]

Example

Give the serial interface to IBASIC.

DIAG:COMM:SER IBAS
DIAG:BOOT:WARM

*Note: :OWNer is implied
Complete the allocation*

**:COMMunicate
:SERial[0][:OWNer]?**

DIAGnostic:COMMunicate:SERial[0][:OWNer]? Returns the current "owner" of the built-in serial interface. The values returned will be; "SYST", "IBAS", or "NONE".

Comments

- **Related Commands:** DIAGnostic:SERial[:OWNer]

Example

Determine which instrument has the serial interface.

DIAG:COMM:SER?
enter statement

*Note: :OWNer is implied
statement returns the string
SYST, IBAS, or NONE*

**:COMMunicate
:SERial[n]:STORE**

DIAGnostic:COMMunicate:SERial[n]:STORE Stores the serial communications parameters (e.g. BAUD, BITS, PARity etc.) into non-volatile storage for the serial interface specified by [n] in SERial[n].

Comments

- Until ...STORE is executed, communication parameter values are stored in *volatile* memory, and a power failure will cause the settings to be lost.
- DIAG:COMM:SER(1-7):STOR causes an HP E1324A (B-size RS-232 card) to store its settings in an on-board EEROM. This EEROM write cycle takes nearly one second to complete. Wait for this operation to complete before attempting to use that serial interface.
- The HP E1324A's EEROM used to store its serial communication settings has a finite lifetime of approximately ten thousand write cycles. Even if your application program sent the ...STORE command once every day, the lifetime of the EEROM would still be over 27 years. Be careful that your application program sends the ...STORE command to an HP E1324A no more often than is necessary.
- **Related Commands:** all SYST:COMM:SER[n]... commands

Example

Store the serial communications settings in the third HP E1324A.

DIAG:COMM:SER3:STOR

**:DOWNload:CHECKed
[:MADDRESS]**

DIAGnostic:CHECKed:DOWNload[:MADDRESS] <address>, <data> writes *data* into a non-volatile User RAM segment starting at *address* using error correction. The User RAM segment is allocated by the DIAG:NRAM:CREate or DIAG:DRAM:CREate command.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFFE)	none
<i>data</i>	arbitrary block program data	See "Parameter Types", in the beginning of this chapter	none

Comments

- This command is typically used to send a block of data to a block of user RAM. It is the only way to send binary data to multiple addresses over a serial (RS232C) line.
- **CAUTION:** Be certain that *all* of the data you download will be contained entirely within the allocated NRAM segment. Writing data outside of the NRAM segment will disrupt the operation of the Command Module. Most computers terminate an OUTPUT, PRINT, or WRITE statement with a carriage return or carriage return and line feed. These End-Of-Line characters must be either accounted for (NRAM segment sized to accommodate them), or suppressed using an appropriate IMAGE or FORMAT statement. Some helpful methods:
 - Size the NRAM segment a little larger than the expected data block
 - Control the End-Of-Line characters with format statements.
 - Use the *Definite Length Arbitrary Block Program Data* format (see example) to send your data rather than the *Indefinite Length Arbitrary Block Program Data* format.
- Address may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats. DOWNload is done by word (16 bits) access so *address* must be even.
- Be certain that *address* specifies a location within the User RAM segment allocated using DIAG:NRAM:CREate if you are downloading a configuration table. DIAG:DOWNload can change the contents of System RAM, causing unpredictable results.
- This command can also be used to write data to a device with registers in the A16 address space. See :DOWNload:SADDRESS.
- **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:ADDRESS?, DIAG:UPLoad?, VXI:CONF:CTABLE, VXI:CONF:DCTable, VXI:CONF:ITABLE, VXI:CONF:MTABLE

DIAGnostic :DOWNload:CHECKed [:MADdress]

Byte Format Each byte sent with this command is expected to be in the following format:

Bit #	7	6	5	4	3	2	1	0
	<i>Control Bit</i>		<i>Check Bits</i>			<i>Data Bits</i>		

- *Control Bit* is used to indicate the serial driver information such as clear, reset, or end of transmission. This bit is ignored by the regular 488.2 driver . The control bit should be one for regular data.
- *Check Bits* are used to detect and correct a single bit error. The control bit is not included in the check. The check bits are a Hamming single bit error correction code, as specified by the following table:

Data Value	Check Bits
0	0
1	7
2	6
3	1
4	5
5	2
6	3
7	4
8	3
9	4
10	5
11	2
12	6
13	1
14	0
15	7

- Data bits are the actual data being transferred (four bits at a time). Each word to be written requires four data bytes for transmission. The significance of the data is dependant on the order received. The first data byte received contains the most significant nibble of the 16 bit word to be written (bits 15-12) . The next data byte received contains the least significant nibble of the most significant byte of the word (bits 11-8). The third data byte received contains the most significant nibble of the least significant byte of the word (bits 7-4). The fourth data byte received contains the least significant nibble of the least significant byte of the word to be written (bits 3-0). Once all four bytes have been received the word will be written.

**:DOWNload:CHECKed
:SADDRESS**

DIAGnostic:CHECKed:DOWNload:SADDRESS < *address* >, < *data* > writes *data* to non-volatile User RAM at a single address specified by *address* using error correction. It can also write to devices with registers in the A16 address space.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFFE)	none
<i>data</i>	arbitrary block program data	See "Parameter Types", in the beginning of this chapter	none

Comments

- This command is typically used to send data to a device which accepts data at a single address. It is the only way to send binary data to single addresses over a serial (RS232C) line.
- Most computers terminate an OUTPUT, PRINT, or WRITE statement with a carriage return or carriage return and line feed. These End-Of-Line characters must be either accounted for (NRAM segment sized to accommodate them), or suppressed using an appropriate IMAGE or FORMAT statement. Some helpful methods:
 - Control the End-Of-Line characters with format statements.
 - Use the *Definite Length Arbitrary Block Program Data* format (see example) to send your data rather than the *Indefinite Length Arbitrary Block Program Data* format.
- A register address in A16 address space can be determined by:

$$1FC00_{16} + (LADDR * 64) + register_number$$

where $1FC00_{16}$ is the base address in the Command Module A16 space, LADDR is the device logical address, 64 is the number of address bytes per device, and register_number is the register to which the data is written.

If the device is an A24 device, the address can be determined using the VXI:CONF:DLIST command to find the base address in A24, and then adding the register_number to that value. A24 memory between address 200000_{16} and address $E00000_{16}$ is directly addressable by the Command Module.

- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats. DOWNload is done by word (16 bit) access so *address* must be even.
- **Related Commands:** DIAG:UPLoad:SADDRESS?

Byte Format Each byte sent with this command is expected to be in the following format:

Bit #	7	6	5	4	3	2	1	0
	<i>Control Bit</i>	<i>Check Bits</i>			<i>Data Bits</i>			

- *Control Bit* is used to indicate the serial driver information such as clear, reset, or end of transmission. This bit is ignored by the regular 488.2 driver. The control bit should be one for regular data.
- *Check Bits* are used to detect and correct a single bit error. The control bit is not included in the check. The check bits are a Hamming single bit error correction code, as specified by the following table:

Data Value	Check Bits
0	0
1	7
2	6
3	1
4	5
5	2
6	3
7	4
8	3
9	4
10	5
11	2
12	6
13	1
14	0
15	7

- *Data bits* are the actual data being transferred (four bits at a time). Each word to be written requires four data bytes for transmission. The significance of the data is dependant on the order received. The first data byte received contains the most significant nibble of the 16 bit word to be written (bits 15-12) . The next data byte received contains the least significant nibble of the most significant byte of the word (bits 11-8). The third data byte received contains the most significant nibble of the least significant byte of the word (bits 7-4). The fourth data byte received contains the least significant nibble of the least significant byte of the word to be written (bits 3-0). Once all four bytes have been received the word will be written.

**:DOWNload
[:MADDRESS]**

DIAGnostic:DOWNload[:MADDRESS] <address>, <data> writes *data* into a non-volatile User RAM segment starting at *address*. The User RAM segment is allocated by the DIAG:NRAM:CREate command.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFFE)	none
<i>data</i>	arbitrary block program data	See "Parameter Types", in the beginning of this chapter	none

Comments

- **CAUTION:** Be certain that *all* of the data you download will be contained entirely within the allocated NRAM segment. Writing data outside of the NRAM segment will disrupt the operation of the Command Module. Most computers terminate an OUTPUT, PRINT, or WRITE statement with a carriage return or carriage return and line feed. These End-Of-Line characters must be either accounted for (NRAM segment sized to accommodate them), or suppressed using an appropriate IMAGE or FORMAT statement. Some helpful methods:
 - Size the NRAM segment a little larger than the expected data block
 - Control the End-Of-Line characters with format statements.
 - Use the *Definite Length Arbitrary Block Program Data* format (see example) to send your data rather than the *Indefinite Length Arbitrary Block Program Data* format.
- This command is generally used to download data into User Configuration Tables. These tables allow the user to control the system's dynamic configuration, interrupt line allocations, commander/servant heirarchy, address space allocation, and mainframe extender configurations.
- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats. DOWNload is done by word (16 bit) access so *address* must be even.
- Be certain that *address* specifies a location within the User RAM segment allocated using DIAG:NRAM:CREate if you are downloading a configuration table. DIAG:DOWNload can change the contents of System RAM, causing unpredictable results.
- This command can also be used to write data to a device with registers in the A16 address space. See :DOWNload:SADDRESS.
- **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:ADDRESS?, DIAG:UPLoad?, VXI:CONF:CTABLE, VXI:CONF:DCTable, VXI:CONF:ITABLE, VXI:CONF:MTABLE

Example Loading Dynamic Configuration information into an allocated RAM segment.

```

DIAG:NRAM:CRE 6           Allocate a segment of user
                           RAM
DIAG:BOOT:WARM           Re-boot system to complete
                           allocation
DIAG:NRAM:ADDR?         query starting address
enter value to variable X get starting address into X
DIAG:DOWN <value of X> ,table data download table data
VXI:CONF:DCTAB <value of X> link configuration table to
                           configuration algorithm
DIAG:BOOT:WARM           Re-boot to set new
                           configuration
    
```

:DOWNload:SADdress

DIAGnostic:DOWNload:SADdress <address>, <data> writes data to non-volatile User RAM at a single address specified by *address*, and writes data to devices with registers in A16 address space.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFFE)	none
<i>data</i>	arbitrary block program data	See "Parameter Types", in the beginning of this chapter	none

Comments

- Most computers terminate an OUTPUT, PRINT, or WRITE statement with a carriage return or carriage return and line feed. These End-Of-Line characters must be accounted for or suppressed using an appropriate IMAGE or FORMAT statement. Some helpful methods:
 - Control the End-Of-Line characters with format statements.
 - Use the *Definite Length Arbitrary Block Program Data* format to send your data rather than the *Indefinite Length Arbitrary Block Program Data* format.
- A register address in A16 address space can be determined by:

$$1FC000_{16} + (LADDR * 64) + register_number$$

where $1FC000_{16}$ is the base address in the Command Module A16 address space, LADDR is the device logical address, 64 is the number of address bytes per device, and register_number is the register to which the data is written.

If the device is an A24 device, the address can be determined using the VXI:CONF:DLIST command to find the base address in A24, and then adding the register_number to that value. A24 memory between address 200000_{16} and address $E00000_{16}$ is directly addressable by the Command Module.

- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats. DOWNload is done by word (16 bit) access so *address* must be even.
- **Related Commands:** DIAG:UPLoad:SADdress?

DIAGnostic:DRAM:AVAIlable?

Example Downloading Data to a Single Address Location

This program downloads an array with the data 1, 2, 3, 4, 5 to register 32 on a device with logical address 40 in VXIbus A16 address space.

```
DIM Dnld_data(1:5) Dimension controller array
DATA 1,2,3,4,5
READ Dnld_data(*) Load data into controller array
OUTPUT "DIAG:DOWN:SADD #H1FCA20,#210";
This line is sent without termination.
Send Dnld_data as 16-bit words Terminate after last word with
EOI or LF and EOI
```

:DRAM:AVAIlable? DIAGnostic:DRAM:AVAIlable? Returns the amount of RAM remaining (available) in the DRAM (Driver RAM) segment, which is the amount of RAM in the segment minus any previously loaded drivers.

- Comments**
- DIAG:DRAM:CREate does not allocate the RAM segment until after a subsequent re-boot.
 - **Related Commands:** DIAG:DRAM:CREate, DIAG:DRIVER:LOAD, DIAG:DRIVER:LIST?

Example Determine amount of space left for drivers in the DRAM segment.

```
DIAG:DRAM:AVA?
enter statement statement returns available
DRAM in bytes.
```

:DRAM:CREate DIAGnostic:DRAM:CREate <size> <num_drivers> creates a non-volatile RAM area for loading instrument drivers. DIAGnostic:DRAM:CREate 0 removes the RAM segment when the system is re-booted.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>size</i>	numeric	0 to available RAM or MIN MAX	none
<i>num_drivers</i>	numeric	0 to available RAM or MIN MAX DEF	8

- Comments**
- *size* is the number of bytes to be allocated to DRAM use. A *size* of zero will remove the DRAM segment.
 - *num_drivers* is the maximum number of drivers to be loaded.
 - The DRAM segment will be created only after the System Instrument has been re-booted (cycle power or execute DIAG:BOOT).
 - Based on the *size* specified, DIAG:DRAM:CRE rounds the *size* up to an even value.

DIAGnostic :DRAM:CREate?

- DRAM will de-allocate previously allocated NRAM and RDISK segments.
- Using all of the available RAM (MAX) for the DRAM segment will limit some functions such as IBASIC program space, instrument reading storage space, and full functionality of the Display Terminal Interface.
- Use DIAG:DRIVER:LOAD... and, DIAG:DRIVER:LIST...? to load and manage DRAM.
- **Related Commands:**DIAG:DRAM:AVailable?, DIAG:DRIVER:LOAD..., DIALG:DRIVER:LIST...?.

Example Allocate a 15 Kbyte non-volatile Driver Ram segment.

DIAG:DRAM:CREate 15360

allocate 15 Kbyte segment of Driver Ram.

:DRAM:CREate? DIAGnostic:DRAM:CREate? [< MIN|MAX> , < MIN|MAX|DEF >] returns the size (in bytes) of a previously created non-volatile RAM area for loading instrument drivers, and the number of drivers currently loaded.

Comments ● If you specify one of the parameters, you must specify both.

:DRIVER:LOAD DIAGnostic:DRIVER:LOAD < driver_block > loads the instrument driver contained in the driver_block into a previously created DRAM segment.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>driver_block</i>	arbitrary block program data	See "Parameter Types" at the beginning of this chapter.	none

Comments

- *driver_block* is the actual binary driver data to be transferred.
- **Related Commands:**DIAG:DRAM:AVailable?, DIAG:DRAM:CREate, DIAG:DRIVER:LIST...?.

Example Download a driver block.

DIAG:DRIV:LOAD

downloads the driver < driver_block > to DRAM memory.

:DRIVER :LOAD: CHECKed DIAGnostic:DRIVER:LOAD:CHECKed <driver_block> loads the instrument driver contained in the driver_block into a previously created DRAM segment. The driver_block is formatted in the same data byte format used by DOWNload:CHECKed.

Parameter Name	Parameter Type	Range of Values	Default Units
driver_block	arbitrary block program data	See "Parameter Types" at the beginning of this chapter.	none

- Comments**
- driver_block is the actual binary driver data to be transferred.
 - This is the only way to download a device driver over a serial (RS232C) line.
 - **Related Commands:**DIAG:DRAM:AVailable?, DIAG:DRAM:CREate, DIAG:DRIVER:LIST...?.

Example Download the driver named DIGITAL.DC.

DIAG:DRIVER:LOAD:CHEC

downloads the driver <driver_block> to DRAM memory.

:DRIVER :LIST[:type]? DIAGnostic:DRIVER:LIST[:type]? lists all drivers from the specified table found on the system. If no parameter is specified, all driver tables are searched and the data from each driver table is separated from the others by a semicolon.

Parameter Name	Parameter Type	Range of Values	Default Units
type	discrete	ALL RAM ROM	ALL

For each driver listed, the following items are returned:

NAME, IDN_MODEL, REV_CODE, TABLE

Parameter	Description
NAME	The instrument name. This is the same label that appears on the instrument selection menu.
IDN_MODEL	The model name. This is the same model name as used in the response to the *IDN? command.
REV_CODE	The revision code. It is in the form A.nn.nn where A as an alpha character
TABLE	The name of the table the driver was found in. This will be RAM or ROM.

- Comments**
- **DIAGnostic:DRIVER:LIST?** lists all drivers found in the system.
 - **DIAGnostic:DRIVER:LIST:RAM?** lists all drivers found in the RAM driver table DRAM.

DIAGnostic :INTerrupt:ACTivate

- **DIAGnostic:DRIVer:LIST:ROM?** lists all drivers found in the ROM driver table.
- **Related Commands:**DIAG:DRAM:AVAILable?, DIAG:DRAM:CREate, DIAG:DRIVer:LOAD...

Example List all drivers in the system.

DIAG:DRIV:LIST? *lists all drivers currently loaded.*

Example List all drivers in ROM.

DIAG:DRIV:LIST:ROM? *lists all of the drivers currently loaded in ROM.*

:INTerrupt:ACTivate

DIAGnostic:INTerrupt:ACTivate < mode > enables an interrupt on the VXI backplane interrupt line specified by **DIAG:INT:SET[n]** to be acknowledged.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>mode</i>	boolean	0 1 OFF ON	none

Comments

- When an interrupt occurs and has been acknowledged, the response is read with the **DIAGnostic:INTerrupt:RESPonse?** command.
- If an interrupt occurs on a VXIbus backplane interrupt line and the interrupt acknowledgement has not been enabled, there is no interrupt acknowledgement response. The interrupt will be held off until the interrupt acknowledge is enabled by either the **DIAG:INT:ACT** command or **DIAG:INT:RESP?** command.
- ON or 1 enable interrupt acknowledgement. OFF or 0 disables interrupt acknowledgement.
- In order for an interrupt to be serviced using the **DIAGnostic:INTerrupt** commands, the interrupt line [n] must be assigned to an interrupt handler using the Interrupt Line Allocation table covered in Chapter 2.
- Bit 8 in the Operation Status register can be used to indicate when an interrupt has been acknowledged (see chapter 4 for details).
- **Related Commands:** **DIAG:INT:PRIority[n]**, **DIAG:INT:RESP?**, **DIAG:INT:SET[n]**
- ***RST Condition:** **DIAG:INT:ACTivate OFF** (for all lines)
- Interrupt acknowledgment must be re-enabled each time an interrupt is acknowledged.

Example Enable an Interrupt Acknowledgement on Line 2.

DIAG:INT:SET2 *Set up interrupt line 2*
DIAG:INT:ACT ON *Enable interrupt to be acknowledged*

:INTerrupt:SETup < n >

DIAGnostic:INTerrupt:SETup < n > < mode > specifies that an interrupt on VXI backplane interrupt line [n] will be serviced by the System Instrument service routine (DIAGnostic:INTerrupt commands) rather than the operating system service routine.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>mode</i>	boolean	0 1 OFF ON	none

Comments

- ...SETup1 through ...SETup7 specify the VXI interrupt lines 1 through 7.
- Sending SETup without an [n] value specifies VXI interrupt line 1.
- ON or 1 specify that interrupt handling is to be set up for the specified interrupt line. OFF or 0 indicate that interrupt handling of the specified line is to be done by the operating system.
- In order for an interrupt to be serviced using the DIAGnostic:INTerrupt commands, the interrupt line [n] must be assigned to an interrupt handler using the Interrupt Line Allocation table covered in Chapter 2.
- **Related Commands:** DIAG:INT:ACT, DIAG:INT:PRiority[n], DIAG:INT:RESP?
- ***RST Condition:** DIAG:INT:SETup[n] OFF (for all lines)

Example

Setup and wait for VXI interrupt response on line 2.

```

DIAG:INT:PRI2 5           set priority to 5 on line 2
DIAG:INT:SETUP2 ON       handle interrupt on line 2
.                          code which will
.                          initiate an action
.                          resulting in an interrupt
DIAG:INT:RESP?          Read the acknowledge response
    
```

:INTerrupt:SETup < n > ?

DIAGnostic:INTerrupt:SETup < n > ? Returns the current state set by DIAG:INT:SETUP[n] < mode >, for the VXI interrupt line specified by [n] in ...SETup[n]?

Comments

- ...SETup1? through ...SETup7? specify the VXI interrupt lines 1 through 7.
- Sending SETup? without an [n] value specifies VXI interrupt line 1.
- If 1 is returned, interrupt handling is set up for the specified interrupt line using the System Instrument (DIAGnostic:INTerrupt commands). If 0 is returned, interrupt handling is done by the operating system.
- **Related Commands:** DIAG:INT:SETup[n], DIAG:INT:PRiority[n], DIAG:INT:ACT, DIAG:INT:RESP?

Example Determine interrupt setup for line 4.

DIAG:INT:SETUP4?

enter statement

statement returns 0 or 1

:INTerrupt:PRiority < n > **DIAGnostic:INTerrupt:PRiority < n >** [*< level >*] gives a priority level to the VXI interrupt line specified by [n] in ...PRiority[n].

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>level</i>	numeric	1 through 7 MIN MAX DEF	none

Comments

- The priority of an interrupt line determines which line will be acknowledged first when more than one line is interrupting.
- For *level*, lower values have lower priority (level 1 is lower priority than level 2).
- No parameter, or DEF (default) sets priority to 1.
- ...PRiority1 through ...PRiority7 specify the VXI interrupt lines 1 through 7.
- Sending PRiority without an [n] value specifies VXI interrupt line 1.
- In order for an interrupt to be serviced using the DIAGnostic:INTerrupt commands, the interrupt line [n] must be assigned to an interrupt handler using the Interrupt Line Allocation table covered in Chapter 2.
- This command has no effect if only one interrupt line is to be set up.
- **Related Commands:** DIAG:INT:ACT, DIAG:INT:SETup[n], DIAG:INT:RESP?

Example Setup, set a priority, and wait for VXI interrupt response on line 2.

DIAG:INT:PRI2 5

handle interrupt on line 2

DIAG:INT:PRI2 5

set priority to 5 on line 2

*

code which will

*

initiate an action

*

resulting in an interrupt

DIAG:INT:RESP?

Read the acknowledge response

:INTerrupt :PRiority < n > ? **DIAGnostic:INTerrupt:PRiority < n > ?** Returns the current priority level set for the VXI interrupt line specified by [n] in ...PRiority[n]?

- Comments**
- ...PRiority?1 through ...PRiority?7 specify the VXI interrupt lines 1 through 7.
 - Sending PRiority? without an [n] value specifies VXI interrupt line 1.
 - **Related Commands:** DIAG:INT:PRiority[n], DIAG:INT:SETup[n], DIAG:INT:RESP?

Example Determine interrupt priority for line 4.

DIAG:INT:PRI4?

enter statement

statement returns 1 through 7

:INTerrupt:RESPonse? **DIAGnostic:INTerrupt:RESPonse?** Returns the interrupt acknowledge response (STATUS/ID word) from the highest priority VXI interrupt line.

- Comments**
- The value returned is the response from the interrupt acknowledge cycle (STATUS/ID word) of a device interrupting on one of the interrupt lines set up with the DIAG:INT:SET[n] command.
 - Bits 0 through 7 of the STATUS/ID word are the interrupting device's logical address. Bits 8 through 15 are Cause/Status bits. Bits 16 through 31 (D32 Extension) are not read by the System Instrument.
 - If only bits 0 through 7 are used by the device (bits 8 - 15 are FF), the logical address can be determined by adding 256 to the value returned by DIAG:INT:RESP?. If bits 0 - 15 are used, the logical address address is determined by adding 65536 to the value returned (if the number returned is negative).
 - Only the interrupt lines previously configured with the DIAG:INT:SET[n] commands generate responses for this command.
 - If there are interrupts on multiple lines when this command is received, or when the acknowledgement was enabled with DIAG:INT:ACT, the response data returned will be from the line with the highest priority set using the DIAG:INT:PRI[n] command.
 - If interrupt acknowledge has not been enabled with DIAG:INT:ACT, then it will be enabled by DIAG:INT:RESP?. System Instrument execution is halted until the interrupt acknowledgement response is received.
 - DIAG:INT:WAIT? can also be used to wait for the interrupt response.
 - **Related Commands:** DIAG:INT:ACT, DIAG:INT:SETup[n], DIAG:INT:PRiority[n]

Example Setup and wait for VXI interrupt response on line 2.

```

DIAG:INT:PRI2 5           set priority to 5 on line 2
DIAG:INT:SETUP2 ON       handle interrupt on line 2
                           code which will
                           initiate an action
                           resulting in an interrupt
                           read the acknowledge response
DIAG:INT:RESP?
    
```

:NRAM:ADDRess? DIAGnostic:NRAM:ADDRess? Returns the starting address of the non-volatile User RAM segment allocated using DIAG:NRAM:CREate.

- Comments**
- DIAG:NRAM:CREate does not allocate the RAM segment until after a subsequent re-boot. To get accurate results, execute DIAG:NRAM:ADDRess? after the re-boot.
 - **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:CREate?, DIAG:DOWNload, DIAG:UPload?

Example Determine address of the most recently created User RAM segment

```

DIAG:NRAM:ADDR?
enter statement           statement returns decimal
                           numeric address
    
```

:NRAM:CREate DIAGnostic:NRAM:CREate <size> allocates a segment of non-volatile User RAM for a user-defined table.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>size</i>	numeric	0 to available RAM or MIN MAX	none

- Comments**
- The RAM segment will be created only after the System Instrument has been re-booted (cycle power or execute DIAG:BOOT).
 - Based on the *size* specified, DIAG:NRAM:CRE rounds the *size* up to an even value.
 - NRAM will de-allocate a previously allocated RDISk segment.
 - Using all of the available RAM (MAX) for the NRAM segment will limit some functions such as IBASIC program space, instrument reading storage space, and full functionality of the Display Terminal Interface.
 - Use DIAG:NRAM:ADDR? to determine the starting address of the RAM segment.
 - Use DIAG:DOWNload, DIAG:UPload?, DIAG:PEEK, or DIAG:POKE to store and retrieve information in the non-volatile RAM segment.
 - Use DIAG:NRAM:CRE? MAX to find maximum available segment size.

DIAGnostic:NRAM:CREate?

- **Related Commands:** DIAG:NRAM:CREate?, DIAG:NRAM:ADDRess?, DIAG:DOWNload, DIAG:UPLoad?

Example Allocate a 15 Kbyte User Non-volatile Ram segment.

DIAG:NRAM:CREate 15360 *allocate 15 Kbyte segment of User Ram.*

:NRAM:CREate? DIAGnostic:NRAM:CREate? [MIN | MAX] Returns the current or allowable (MIN | MAX) size of the User non-volatile RAM segment.

- Comments**
- DIAG:NRAM:CRE does not allocate driver RAM until a subsequent re-boot. To get accurate results, execute DIAG:NRAM:CRE? after the re-boot.
 - **Related Commands:** DIAG:NRAM:ADDRess?, DIAG:NRAM:CREate

Example Check the size of the User RAM segment.

DIAG:NRAM:CREate?
enter statement *statement enters size in bytes*

:PEEK? DIAGnostic:PEEK? <address>, <width> reads the data (number of bits given by *width*) starting at *address*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFF)	none
<i>width</i>	numeric	8 16 32	none

- Comments**
- *Address* specifies a location within the range of the control processor's addressing capability.
 - *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
 - **Related Commands:** DIAG:POKE

Example Read byte from User non-volatile RAM

DIAG:PEEK? 16252928,8 *ask for byte*
enter statement *return value of byte*

:POKE **DIAGnostic:POKE** <address>, <width>, <data> writes *data* (number of bits given by *width*) starting at *address*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFF)	none
<i>width</i>	numeric	8 16 32	none
<i>data</i>	numeric	8 to 32 bit integer	none

Comments

- *Address* specifies a location within the range of the control processor's addressing capability.
- Address and *data* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- **CAUTION:** DIAG:POKE can change the contents of any address in RAM. Changing the contents of RAM used by the Command Module's control processor can cause unpredictable results.
- **Related Commands:** DIAG:PEEK?

Example

Store byte in User non-volatile RAM

DIAG:POKE 16252928,8,255

:RDISK:ADDRESS?

DIAGnostic:RDISK:ADDRESS? Returns the starting address of the RAM disc volume previously defined with the DIAG:RDISK:CREate command. The RAM disc volume is defined for use only by the IBASIC option.

Comments

- DIAG:RDISK:CREate does not allocate the RAM volume segment until after a subsequent re-boot. To get accurate results, execute DIAG:RDISK:ADDRESS? after the re-boot.
- **Related Commands:** DIAG:RDISK:CREate, DIAG:RDISK:CREate?

Example

Return the starting address of the IBASIC RAM volume.

DIAG:RDIS:ADDR?

enter statement

statement returns decimal numeric address

:RDISk:CREate **DIAGnostic:RDISk:CREate** <*size*> Allocates memory for a RAM disc volume. The RAM disc volume is defined for use only by the IBASIC option.

Parameter Name	Parameter Type	Range of Values	Default Units
<i>size</i>	numeric	0 to available RAM or MIN MAX	none

- Comments**
- The RAM disc segment will only be created after the System Instrument has been re-booted (cycle power or execute DIAG:BOOT).
 - Using all of the available RAM (MAX) for the disc volume segment will limit some functions such as IBASIC program space, instrument reading storage space, and full functionality of the Display Terminal Interface.
 - **Related Commands:** DIAG:RDISk:ADDress?, DIAG:RDISk:CREate?

Example Allocate a 64 Kbyte segment for the IBASIC option's RAM volume.
DIAG:RDIS:CRE 65536

:RDISk:CREate? **DIAGnostic:RDISk:CREate?** [MIN | MAX] Returns the current or allowable (MIN | MAX) size of the RAM disc volume segment.

- Comments**
- DIAG:RDIS:CRE does not allocate driver RAM until a subsequent re-boot. To get accurate results, execute DIAG:RDIS:CRE? after the re-boot.
 - **Related Commands:** DIAG:RDISk:CREate, DIAG:RDISk:ADDR?

Example Return the size of the current RAM disc volume.
DIAG:RDIS:CRE?
 enter statement *returns numeric size*

:UPLoad[:MADdress]?

DIAGnostic:UPLoad[:MADdress]? <address>, <byte_count> Returns the number of bytes specified by *byte_count*, starting at *address*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFFE)	none
<i>byte_count</i>	numeric	0 to (999,999,998)	none

Comments

- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- UPLoad is done by word (16 bit) access so *address* and *byte_count* must be even.
- Data is returned in the Definite Block Response Data format:

< non-zero digit > < digit(s) > < data byte(s) >

Where the value of < non-zero digit > equals the number of < digit(s) >. The value of < digit(s) > taken as a decimal integer indicates the number of < data byte(s) > to expect in the block.
- This command can also be used to retrieve data from a device with registers in A16 address space. See DIAG:UPLoad:SADdress?
- Related Commands: DIAG:NRAM:ADdress?, DIAG:NRAM:CREate, DIAG:DOWNload

Example Upload data stored on non-volatile User RAM.

```

DIM HEADER$(6),DATA(1024)
    6 chars for "#41024" header
    1024 chars for data bytes
DIAG:NRAM:ADDR?
    get starting address of NRAM
enter ADD
    address into ADD
OUTPUT "DIAG:UPL? < value of ADD >,1024"
    request 1 Kbyte from address in ADD
enter HEADER$
    strip "#41024" from data
enter DATA
    get 1024 data bytes into string; use enter format so statement
    won't terminate on CRs or LFs etc. Line Feed (LF) and EOI
    follow the last character retrieved.
    
```

:UPload:SADdress?

DIAGnostic:UPload:SADdress? <address>, <byte_count> Returns the number of bytes specified by *byte_count*, at *address*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>address</i>	numeric	0 to 16,777,215 (#HFFFFFFE)	none
<i>byte_count</i>	numeric	0 to (999,999,998)	none

Comments

- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- ULLoad is done by word (16 bit) access so *address* and *byte_count* must be even.
- The register address in A16 address space can be determined by:
 $1FC000_{16} + (LADDR * 64) + register_number$
 where $1FC000_{16}$ is the base address in the VXIbus A16 address space, LADDR is the device logical address, 64 is the number of address bytes per device, and *register_number* is the register from which data is retrieved.

 If the device is an A24 device, the address can be determined using the VXI:CONF:DLIST command to find the base address in A24, and then adding the *register_number* to that value. A24 memory between address 200000_{16} and address $E00000_{16}$ is directly addressable by the Command Module.
- Data is returned in the Definite Block Response Data format:
 $\# < non\text{-}zero\ digit > < digit(s) > < data\ byte(s) >$
 Where the value of < non-zero digit > equals the number of < digit(s) >. The value of < digit(s) > taken as a decimal integer indicates the number of < data byte(s) > to expect in the block.
- **Related Commands:** DIAG:DOWNload:SADdress

Example Upload data stored in non-volatile User RAM.

This program reads 1024 data bytes from register 32 on a device with logical address 40 in Command Module A16 address space.

```

DIM HEADER$[6],DATA(1024)           6 chars for "#41024" header
                                     1024 chars for data bytes
DIAG:UPL:SADD? #H1FCA20,1024       request 1 Kbyte from device
                                     register 32
enter HEADER$                       strip "#41024" from data
enter DATA
    
```

get 1024 data bytes into string; use enter format so statement won't terminate on CRs or LFs etc. Line Feed (LF) and EOI follow the last character retrieved.

OUTPut

The OUTPut subsystem controls the output of pulses and levels to the ECLTRG, and TTLTRG* trigger buses as well as the module's front panel "Trig Out" connector. Signals connected to the front panel "Trig In" connector can also operate the ECLTRG and TTLTRG* trigger buses.

NOTE

The ECLTrg trigger lines, TTLTrg trigger lines, and the Command Module "Trig Out" port use "low true" or negative logic. When a trigger level is set (e.g. OUTP:EXT:LEV 1), a low voltage is present.

Subsystem Syntax

```

OUTPut
  :ECLTrg < n > (:ECLTrg0 or :ECLTrg1)
    :IMMediate
    :LEVel
      [:IMMediate] < level >
    :LEVel?
    :SOURce < source >
    :SOURce?
    [:STATe] < state >
    :STATe?
  :EXTErnal
    :IMMediate
    :LEVel
      [:IMMediate] < level >
    :LEVel?
    :SOURce < source >
    :SOURce?
    :STATe < state >
    :STATe?
  :TTLTrg < n > (:TTLTrg0 through :TTLTrg7)
    :IMMediate
    :LEVel
      [:IMMediate] < level >
    :LEVel?
    :SOURce < source >
    :SOURce?
    [:STATe] < state >
    :STATe?
  
```

:ECLTrg < n > :IMMediate **OUTPut:ECLTrg < n > :IMMediate** causes a pulse to appear on the specified ECL trigger line.

Comments

- ECLTrg < n > represents either ECLTrg0 or ECLTrg1.
- OUTP:ECLT < n > :STATE must be ON and OUTP:ECLT < n > :SOUR must be set to INT or NONE in order to issue an immediate pulse. A "settings conflict" error is generated if STATE is not on.
- **Related Commands:** OUTP:ECLT:SOUR, OUTP:ECLT:STAT

Example Send trigger pulse to ECLTRG0.

```

OUTP:ECLT0:STATE ON
OUTP:ECLT0:SOUR INT
OUTP:ECLT0:IMM
  
```

*Set System Instrument to send a pulse on ECLT0
set trigger source to internal pulse the ECLTRG0 bus*

OUTPut:ECLTrg <n> :LEVel[:IMMediate]

:ECLTrg <n> :LEVel[:IMMediate]

OUTPut:ECLT<n>:LEV[:IMMediate] <level> sets the selected ECLTRG trigger line to logical level 0 or 1.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>level</i>	boolean	0 1 OFF ON	none

Comments

- ECLTrg<n> represents either ECLTrg0 or ECLTrg1.
- OUTP:ECLT<n>:STATE must be ON
- OUTP:ECLT<n>:SOURCE must be INTERNAL.
- OUTP:ECLT<n>:STATE must be ON for the source to drive the trigger line. Setting . . . :STATE OFF does not change the source, so the signal driving the line is still present. Setting . . . :STATE back ON sets the source to NONE and de-asserts the line.
- **Related Commands:** OUTP:ECLT<n>:LEV?, OUTP:ECLT<n>:SOUR, OUTP:ECLT<n>:STAT
- ***RST Condition:** OUTP:ECLT<n>:LEV 0

Example ECLTRG0 set to logic level 1

```

OUTP:ECLT0:STATE ON           Enable ECLT0
OUTP:ECLT0:SOUR INT          Set the source to internal
OUTP:ECLT0:LEV 1             Set trigger level
    
```

:ECLTrg <n> :LEVel[:IMMediate]?

OUTPut:ECLT<n>:LEVel[:IMMediate]? returns the current logic level of the selected ECLTRG trigger line.

Example Determine current state of ECLTRG1

```

OUTP:ECLT1:LEV?              ask for level
enter statement               return state of trigger line.
    
```

:ECLTrg <n> :SOURCE

OUTPut:ECLTrg<n>:SOURCE <parameter> selects which source will drive the selected trigger line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>source</i>	discrete	INT EXT NONE	none

Comments

- ECLTrg<n> represents either ECLTrg0 or ECLTrg1.
- INTERNAL allows the selected trigger line to be driven by OUTP:ECLT<n>:LEV commands.

OUTPut :ECLTrg <n> :SOURce?

- EXTERNAL allows the selected trigger line to be driven by the "Trig In" front panel BNC connector.
- OUTP:ECLT <n> :STATE must be ON for the source to drive the trigger line. Setting . . . :STATE OFF does not change the source, so the signal driving the line is still present. Setting . . . :STATE back ON sets the source to NONE and de-asserts the line.
- **Related Commands:** OUTP:ECLT <n> :STAT, OUTP:ECLT <n> :LEV
- ***RST Condition:** OUTP:ECLT <n> :SOUR NONE

Example Select the TRIG IN connector to drive ECLTRG0

OUTP:ECLT0:SOUR EXT

:ECLTrg <n> :SOURce?

OUTPut:ECLTrg[n]:SOURce? queries the source currently driving the selected trigger line.

Comments

- Querying the source with . . . :STATE OFF returns NONE, regardless of the actual source setting.

Example

Determine the source driving ECLTRG1

OUTP:ECLT1:SOUR?

enter statement

return trigger source

:ECLTrg <n> [:STATE]

OUTPut:ECLTrg <n> [:STATE] <state> enables configuration (e.g. source, level) of the specified trigger line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>state</i>	boolean	0 1 OFF ON	none

Comments

- ECLTrg <n> represents either ECLTrg0 or ECLTrg1.
- When a trigger line is asserted (OUTP:ECLT <n> :LEV 1), it remains asserted when . . . :STATE OFF it set. Setting . . . :STATE ON again de-asserts the line by setting the source to NONE.
- **Related Commands:** OUTP:ECLT <n> :LEV, OUTP:ECLT <n> :SOUR
- ***RST Condition:** OUTP:ECLT <n> :STAT 0

Example

Enable the ECLTRG1 trigger bus

OUTP:ECLT1:STATE ON

OUTPut:ECLTrg <n> [:STATe]?

:ECLTrg <n> [:STATe]? **OUTPut:ECLTrg <n> [:STATe]?** returns the current state (ON or OFF) of the selected trigger line.

Example Query for the state of ECLTRG1

OUTP:ECLT1:STAT?
enter statement *return the current state*

:EXTErnal:IMMEdiate **OUTPut:EXTErnal:IMMEdiate** causes a pulse to appear on the front panel "Trig Out" port.

- Comments**
- OUTP:EXT:STATE must be ON and OUTP:EXT:SOUR must be INT or NONE.
 - **Related Commands:** OUTP:EXT:STAT, OUTP:EXT:SOUR

Example Send trigger pulse to Trig Out port.

OUTP:EXT:STAT ON *Enable Trig Out port*
OUTP:EXT:SOUR INT *Set trigger source*
OUTP:EXT:IMM *pulse Trig Out*

:EXTErnal :LEVEl[:IMMEdiate] **OUTPut:EXT:LEV <level>** sets the "Trig Out" port to a logical level 0 or 1.

Parameter Name	Parameter Type	Range of Values	Default Units
<i>level</i>	boolean	0 1 OFF ON	none

- Comments**
- OUTP:EXT:STATE must be ON.
 - OUTP:EXT:SOURCE must be INTERNAL
 - Once the level of the Trig Out port is set to logical level 1, it remains set if OUTP:EXT:STATE OFF is set. Setting OUTP:EXT:STATE back to ON sets the output back to logical level 0, and sets OUTP:EXT:SOUR to NONE.
 - **Related Commands:** OUTP:EXT:LEV?, OUTP:EXT:SOUR, OUTP:EXT:STAT
 - ***RST Condition:** OUTP:EXT:LEV 0

Example Set Trig Out port to a logic level 1

OUTP:EXT:STATE ON *Enable output*
OUTP:EXT:SOUR INT *Set trigger source internal*
OUTP:EXT:LEV 1 *Set output level*

:EXtErnal :LEVel[:IMMediate]?

OUTPut:EXtErnal:LEVel[:IMMediate]? returns the current logic level of the "Trig Out" port.

Example Determine current state of Trig Out port

OUTP:EXT:LEV?
enter statement *ask for level*
return state of trigger bus.

:EXtErnal :SOURce

OUTPut:EXtErnal:SOURce <parameter> selects which source will drive the "Trig Out" port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
source	discrete	INT TTLT <n> ECLT <n> NONE	none

Comments

- INTERNAL allows the "Trig Out" port to be driven by OUTP:EXT:LEV.
- TTLTrg<n> or ECLTrg<n> allows the "Trig Out" port to be driven by the selected VXIbus trigger line.
- OUTP:EXT:STATE must be ON for the source to operate the "Trig Out" port. Setting . . . :STATE OFF does not change the source, so the signal driving the port is still present. Setting . . . :STATE back ON sets the source to NONE.
- **Related Commands:** OUTP:EXT:STAT, OUTP:EXT:LEV
- ***RST Condition:** OUTP:EXT:SOUR NONE

Example Select TTLTRG0* to drive the Trig Out port

OUTP:EXT:SOUR TTLT0

:EXtErnal :SOURce?

OUTPut:EXtErnal:SOURce? queries for the source currently driving the "Trig Out" port.

Comments

- Querying the source with . . . :STATE OFF returns NONE, regardless of the actual source setting.

Example Determine the source driving "Trig Out"

OUTP:EXT:SOUR?
enter statement *return Trig Out source*

:EXtErnal [:STATe] **OUTPut:EXtErnal[:STATe]** <state> enables configuration (e.g. source, level) of the Command Module's "Trig Out" port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
state	boolean	0 1 OFF ON	none

Comments

- When the Trig Out port is set to logical level 1, it remains set if **OUTP:EXT:STATE** is set to OFF. Setting **OUTP:EXT:STATE** back to ON sets the Trig Out port back to logical level 0. **OUTP:EXT:SOUR** is set to NONE.
- **Related Commands:** **OUTP:EXT:SOUR**, **OUTP:EXT:LEV**
- ***RST Condition:** **OUTP:EXT:STAT 0**

Example

Enable the "Trig Out" port
OUTP:EXT:STATE ON

:EXtErnal[:STATe]? **OUTPut:EXtErnal[:STATe]?** returns the current state (ON or OFF) of the "Trig Out" port.

Example

Query for the state of "Trig Out"
OUTP:EXT:STAT?
 enter statement *return the current state*

:TTLTrg <n> :IMMediate:IMM **OUTPut:TTLTrg <n> :IMMediate** causes a pulse to appear on the specified TTL trigger line.

Comments

- **TTLTrg <n>** represents **TTLTrg0** through **TTLTrg7**.
- **OUTP:TTLTRG <n> :STATE** must be ON and **OUTP:TTLTRG <n> :SOUR** must be set to INT or NONE in order to issue an immediate pulse. A "settings conflict" error is generated if STATE is not on.
- **Related Commands:** **OUTP:TTLT:SOUR**, **OUTP:TTLT:STAT**

Example

Send trigger pulse to **TTLTRG0*** and **TTLTRG4***.

OUTP:TTLT0:STAT ON *Enable the System Instrument to send a pulse on TTLT0 and TTLT4*
OUTP:TTLT4:STAT ON

OUTP:TTLT0:SOUR INT
OUTP:TTLT4:SOUR INT *Set trigger sources*
OUTP:TTLT0:IMM *pulse the TTLTRG0 bus*
OUTP:TTLT4:IMM *pulse the TTLTRG4 bus*

**:TTLTrg <n>
:LEVel[:IMMediate]**

OUTPut:TTLT<n>:LEV <level> sets the selected TTLTRG* trigger line to logical level 0 or 1.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
level	boolean	0 1 OFF ON	none

Comments

- TTLTrg<n> represents TTLTrg0 through TTLTrg7.
- OUTP:TTLT<n>:STATE must be ON for the source to drive the trigger line. Setting . . . :STATE OFF does not change the source, so the signal driving the line is still present. Setting . . . :STATE back ON sets the source to NONE and de-asserts the line.
- OUTP:TTLT<n>:STATE must be ON.
- OUTP:TTLT<n>:SOURCE must be INTERNAL.
- **Related Commands:** OUTP:TTLT<n>:LEV?, OUTP:TTLT<n>:SOUR, OUTP:TTLT<n>:STAT
- ***RST Condition:** OUTP:TTLT<n>:LEV 0

Example TTLTRG0* set to logic level 1

```

OUTP:TTLT0:STAT ON           Enable TTLT0
OUTP:TTLT0:SOUR INT         Set source to internal
OUTP:TTLT0:LEV 1           Set trigger level
    
```

**:TTLTrg <n>
:LEVel[:IMMediate]?**

OUTPut:TTLT<n>:LEVel[:IMMediate]? returns the current logic level of the TTLTRG* line specified by [n].

Example Determine current state of TTLTRG1*

```

OUTP:TTLT1:LEV?           ask for level
enter statement          return state of trigger line.
    
```

:TTLTrg <n> :SOURCE

OUTPut:TTLTrg<n>:SOURCE <source> selects which source will drive the selected trigger line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
source	discrete	INT EXT NONE	none

Comments

- TTLTrg<n> represents TTLTrg0 through TTLTrg7.
- INTERNAL allows the selected trigger line to be driven by OUTP:TTLT<n>:LEV.

STATus :TTLTrg <n> [:STATe]?

Example Enable the TTLTRG1* trigger line

OUTP:TTL1:STAT ON

:TTLTrg <n> [:STATe]?

OUTPut:TTLTrg <n> [:STATe]? returns the current state (ON or OFF) of the selected trigger line.

Example Query for the state of TTLTRG1*

OUTP:TTL1:STAT?

enter statement

return the current state

STATus

The STATus subsystem commands access the condition, event, and enable registers in the Operation Status Group and the Questionable Data Group.

Subsystem Syntax

STATus

:OPERation

:CONDition?

:ENABle <event >

:ENABle?

[:EVENTt]?

:PRESet

:QUEStionable

:CONDition?

:ENABle <event >

:ENABle?

[:EVENTt]?

:OPERation

:CONDition?

STATus:OPER:COND? returns the state of the condition register in the Operation Status Group. The state represents conditions which are part of an instrument's operation.

Comments

- Bit 8 in the register is used by the System Instrument (Command Module) to indicate when an interrupt set up by the DIAG:INTerrupt commands has been acknowledged.
- Reading the condition register does not change the setting of bit 8. Bit 8 is cleared by the DIAG:INT:RESP? command.
- **Related Commands:** STAT:OPER:ENABle, STAT:OPER:EVENT?

Example

Reading the contents of the condition register

STAT:OPER:COND?

enter statement

query register

:OPERation:ENABLE

STATus:OPER:ENABLE <event> sets an enable mask to allow events monitored by the condition register and recorded in the event register, to send a summary bit to the Status Byte register (bit 7).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
event	numeric	256	none

Comments

- Bit 8 in the condition register is used by the system instrument (Command Module) to indicate when an interrupt set up by the DIAG:INTerrupt commands has been acknowledged.
- Bit 8 is the only bit used in the condition register (by the System Instrument), therefore, it is the only bit which needs to be unmasked in the event register. Specifying the "bit weight" for the event unmask the bit. The bit weight is 256 and can be specified in decimal, hexadecimal (#H), Octal (#Q) or binary (#B).
- When the summary bit is sent, it sets bit 7 in the Status Byte register.
- **Related Commands:** STAT:OPER:ENABLE?

Example

Unmasking bit 8 in the Event Register

STAT:OPER:ENAB 256 *unmask bit 8*

:OPERation:ENABLE?

STATus:OPER:ENABLE? returns which bits in the event register (Operation Status Group) are unmasked.

Comments

- Bit 8 in the condition register is used by the system instrument (Command Module) to indicate when an interrupt set up by the DIAG:INTerrupt commands has been acknowledged.
- Bit 8 in the event register generally is the only bit which will be unmasked. If this bit is unmasked when STAT:OPER:ENAB? is sent, 256 is returned.
- Reading the event register mask does not change the mask setting (STAT:OPER:ENAB <event>).
- **Related Commands:** STAT:OPER:ENABLE

Example

Reading the Event Register Mask

STAT:OPER:ENAB?
enter statement *query register mask*

:OPERation[:EVENT]?

STATus:OPER:EVENT? returns which bits in the event register (Operation Status Group) are set. The event register indicates when there has been a positive transition in the condition register.

Comments

- Bit 8 in the condition register is used by the system instrument (Command Module) to indicate when an interrupt set up by the DIAG:INTerrupt commands has been acknowledged.
- Bit 8 in the event register generally is the only bit which is used. If this bit is set when STAT:OPER:EVENT? is sent, 256 is returned.
- Reading the event register clears the contents of the register. If the event register is to be used to generate a service request (SRQ), you should clear the register before enabling the SRQ (*SRE). This prevents an SRQ from occurring due to a previous event.
- **Related Commands:** STAT:OPER:ENABLE, STAT:OPER:ENABLE?

Example **Reading the Event Register**

```
STAT:OPER:EVENT?           query if bit(s) is set
enter statement
```

:PRESet

STATus:PRESet sets each bit in the enable register (standard operation status group) to '0'.

Example **Presetting the Enable Register**

```
STAT:PRES                 preset enable register
```

:QUESTionable

The STATus:QUESTionable commands are supported by the system instrument, however, they are not used by the System Instrument. Queries of the Questionable Data condition and event registers will always return +0.

SYSTem

The SYSTem command subsystem for the System Instrument provides for:

- Configuration of the RS-232 interface
- Control and access of the System Instrument's real time clock/calendar (SYST:TIME, SYST:TIME?, SYST:DATE, SYST:DATE?).
- Access to the System Instrument's error queue (SYST:ERR?).
- Configuring the communication ports (HP-IB and serial).

Subsystem Syntax

```

SYSTem
:COMMunicate
  :GPIB
    :ADDRess?
  :SERial[n]
    :CONTRol
      :DTR ON | OFF | STANdard | IBFull
      :DTR?
      :RTS ON | OFF | STANdard | IBFull
      :RTS?
    [:RECeive]
      :BAUD <baud_rate> | MIN | MAX
      :BAUD? [MIN | MAX]
      :BITS 7 | 8 | MIN | MAX
      :BITS? [MIN | MAX]
      :PACE
        [:PROTOcol] XON | NONE
        [:PROTOcol]?
        :THReshold
          :STARt <characters> | MIN | MAX
          :STARt? [MIN | MAX]
          :STOP <characters> | MIN | MAX
          :STOP? [MIN | MAX]
      :PARity
        :CHECK 1 | 0 | ON | OFF
        :CHECK?
        [:TYPE] EVEN | ODD | ZERO | ONE | NONE
        [:TYPE]?
        :SBITs 1 | 2 | MIN | MAX
        :SBITs? [MIN | MAX]
    :TRANsmit
      :AUTO 1 | 0 | ON | OFF
      :AUTO?
      :PACE
        [:PROTOcol] XON | NONE
        [:PROTOcol]?
  :DATE <year>, <month>, <day>
  :DATE? [MIN|MAX,MIN|MAX,MIN|MAX]
  :ERRor?
  :TIME <hour>, <minute>, <second>
  :TIME? [MIN | MAX,MIN | MAX,MIN | MAX]
  :VERsion?
  
```

**:COMMunicate
:GPIB:ADDRess?**

SYSTem:COMMunicate:GPIB:ADDRess? returns the Command Module primary HP-IB address.

Comments

- The HP E1405 Command Module (primary) HP-IB address is set using switches on the module.

Example

Read the Primary HP-IB Address.

SYST:COMM:GPIB:ADDR?
enter statement

Read the HP-IB address
Enter the HP-IB address

**:COMMunicate
:SERial[n]: ...**

The SYSTem:COMMunicate:SERial[n]: ... commands set and/or modify the configuration of the serial interface(s) that are under control of the System Instrument (Command Module). The interface to be affected by the command is specified by a number (zero through seven) which replaces the [n] in the :SERial[n] command. The number is the interface's **card number**. Card number zero specifies the Command Module's built-in interface while one through seven specify one of up to seven E1324 B-size plug-in serial interface modules. The serial interface installed at < System Instrument's logical address > + 1 becomes card number 1, the serial interface installed at the next sequential logical address becomes card number 2 and so on. The logical addresses used by plug-in serial interfaces must start at < System Instrument's logical address > + 1 and be contiguous (no unused logical addresses). The factory set logical address of the HP E1405 Command Module is 0.

Comments

- Serial communication commands take effect *after* the end of the program message containing the command.
- Serial communication settings for the built-in RS-232 interface can be stored in its non-volatile RAM **only** after the DIAG:COMM:SER[n]:STORE command is executed. These settings are used at power-up and DIAG:BOOT[:WARM].
- Serial communication settings for the HP E1324A Datacomm interface can be stored in its on-board non-volatile EEROM **only** after the DIAG:COMM:SER[n]:STORE command is executed. These settings are used at power-up and DIAG:BOOT[:WARM].
- DIAG:BOOT:COLD will set the serial communication parameters to the following defaults:
 - BAUD 9600
 - BITS 8
 - PARity NONE
 - SBITs 1
 - DTR ON
 - RTS ON
 - PACE XON

Example

Setting baud rate for plug-in card 2.

SYST:COMM:SER2:BAUD 9600

(must be a card number 1 also)

**:COMMunicate
:SERial[n] :CONTrol
:DTR**

SYSTem:COMMunicate:SERial[n]:CONTrol:DTR < *dtr_cntrl* > controls the behavior of the Data Terminal Ready output line. DTR can be set to a static state (ON | OFF), can operate as a modem control line (STANDARD), or can be used as a hardware handshake line (IBFull).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>dtr_cntrl</i>	discrete	ON OFF STANDARD IBFull	none

Comments

- The following table defines each value of *dtr_cntrl*:

Value	Definition
ON	DTR line is asserted
OFF	DTR Line is unasserted
STANDARD	DTR will be asserted when the serial interface is ready to send <i>output</i> data. Data will be sent if the connected device asserts DSR and CTS.
IBFull	While the input buffer is not yet at the :STOP threshold, DTR is asserted. When the input buffer reaches the :STOP threshold, DTR will be unasserted.

- DIAG:BOOT:COLD will set ...DTR to ON.
- **Related Commands:** SYST:COMM:SER[n]:CONT:RTS, SYST:COMM:SER[n]:PACE:THR:START, SYST:COMM:SER[n]:PACE:THR:STOP
- *RST Condition: No change

Example Asserting the DTR line.

SYST:COMM:SER0:CONT:DTR ON

**:COMMunicate
:SERial[n] :CONTrol
:DTR?**

SYSTem:COMMunicate:SERial[n]:CONTrol:DTR? returns the current setting for DTR line control.

Example Checking the setting of DTR control.

SYST:COMM:SER0:CONT:DTR?
enter statement

statement enters the string "ON", "OFF", "STAN", or "IBF"

**:COMMunicate
:SERial[n] :CONTrol
:RTS**

SYSTem:COMMunicate:SERial[n]:CONTrol:RTS <Rts_cntrl> controls the behavior of the Request To Send output line. RTS can be set to a static state (ON | OFF), can operate as a modem control line (STANDARD), or can be used as a hardware handshake line (IBFull).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>rts_cntrl</i>	discrete	ON OFF STANDARD IBFull	none

Comments

- The following table defines each value of *rts_cntrl*:

Value	Definition
ON	RTS line is asserted
OFF	RTS Line is unasserted
STANDARD	RTS will be asserted when the serial interface is ready to send <i>output</i> data. Data will be sent if the connected device asserts CTS and DSR.
IBFull	While the input buffer is not yet at the :STOP threshold, RTS is asserted. When the input buffer reaches the :STOP threshold, RTS will be unasserted.

- DIAG:BOOT:COLD will set ...RTS to ON.
- **Related Commands:** SYST:COMM:SER[n]:CONT:DTR, SYST:COMM:SER[n]:PACE:THR:START, SYST:COMM:SER[n]:PACE:THR:STOP
- *RST Condition: No change

Example Unasserting the RTS line.

SYST:COMM:SER0:CONT:RTS OFF

**:COMMunicate
:SERial[n] :CONTrol
:RTS?**

SYSTem:COMMunicate:SERial[n]:CONTrol:RTS? returns the current setting for RTS line control.

Example Checking the setting of RTS control.

SYST:COMM:SER0:CONT:RTS?

enter statement

statement enters the string "ON", "OFF", "STAN", or "IBF"

SYSTem:COMMunicate :SERial[n] [:RECeive] :BAUD

**:COMMunicate
:SERial[n] [:RECeive]
:BAUD**

SYSTem:COMMunicate:SERial[n] [:RECeive]:BAUD <baud_rate> Sets the baud rate for the serial port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>baud</i>	numeric	300 1200 2400 4800 9600 19200 MIN MAX	none

Comments

- Attempting to set *baud* to other than those values shown will result in an error -222.
- DIAG:BOOT:COLD will set ...BAUD to 9600.
- *RST condition: No change.

Example

Setting the baud rate to 1200.

SYST:COMM:SER0:BAUD 1200

**:COMMunicate
:SERial[n] [:RECeive]
:BAUD?**

SYSTem:COMMunicate:SERial[n] [:RECeive]:BAUD? [MIN | MAX] returns:

- The current baud rate setting if no parameter is sent.
- The maximum allowable setting if MAX is sent.
- The minimum allowable setting if MIN is sent.

Example

Querying the current baud rate.

SYST:COMM:SER0:BAUD?

enter statement

statement enters a numeric value

**:COMMunicate
:SERial[n] [:RECeive]
:BITS**

SYSTem:COMMunicate:SERial[n][:RECeive]:BITS <bits> Sets the number of bits to be used to transmit and receive data.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>bits</i>	numeric	7 8 MIN MAX	none

Comments

- Attempting to set *bits* to other than those values shown will result in an error -222.
- While this command operates independently of either the ...PARity:TYPE or ...SBITs commands, there are two combinations which are disallowed because of their data frame bit width. The following table shows the possible combinations:

...BITS	...PARity:TYPE	...SBITs	Frame Bits
7	NONE	1	9 - disallowed
7	NONE	2	10
7	Yes	1	10
7	Yes	2	11
8	NONE	1	10
8	NONE	2	11
8	Yes	1	11
8	Yes	2	12 - disallowed

- DIAG:BOOT:COLD will set ...BITS to 8.
- Related Commands: SYST:COMM:SER[n]:PARity
- *RST Condition: No change

Example Configuring data width to 7 bits.

SYST:COMM:SER0:BITS 7

**:COMMunicate
:SERial[n] [:RECeive]
:BITS?**

SYSTem:COMMunicate:SERial[n][:RECeive]:BITS? [MIN | MAX] returns:

- The current data width if no parameter is sent.
- The maximum allowable setting if MAX is sent.
- The minimum allowable setting if MIN is sent.

Example Querying the current data width.

SYST:COMM:SER0:BITS?

enter statement

statement enters 7 or 8

SYSTem:COMMunicate :SERial[n] [:RECeive] :PACE [:PROToCol]

**:COMMunicate
:SERial[n] [:RECeive]
:PACE [:PROToCol]**

SYSTem:COMMunicate:SERial[n] [:RECeive]:PACE[:PROToCol]
<protocol> enables or disables receive pacing (XON/XOFF) protocol.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>protocol</i>	discrete	XON NONE	none

Comments

- While ...PROT is XON, the serial interface will send XOFF when the buffer reaches the ...STOP threshold, and XON when the buffer reaches the ...START threshold.
- For an HP E1324A, AUTO is always ON. In this case ...[:RECeive]:PACE will also set ...TRAN:PACE
- The XON character is control Q (ASCII 17₁₀, 11₁₆), The XOFF character is control S (ASCII 19₁₀, 13₁₆).
- DIAG:BOOT:COLD will set ...PACE to XON.
- **Related Commands:** ...PROToCol:THReShold:STARt, ...PROToCol:THReShold:STOP, ...TRAN:AUTO
- ***RST Condition:** No change

Example

Enabling XON/XOFF handshaking.
SYST:COMM:SER0:PACE:PROT XON

**:COMMunicate
:SERial[n] [:RECeive]
:PACE [:PROToCol]?**

SYSTem:COMMunicate:SERial[n] [:RECeive]:PACE[:PROToCol]? returns the current receive pacing protocol.

Example

See if XON/XOFF protocol is enabled.
SYST:COMM:SER0:PACE:PROT?
 enter statement *statement enters the string "XON" or "NONE"*

SYSTem :COMMunicate :SERial[n] [:RECeive] :PACE :THReshold :STARt

**:COMMunicate
:SERial[n] [:RECeive]
:PACE :THReshold
:STARt**

SYSTem:COMMunicate:SERial[n] [:RECeive]:PACE:THReshold:STARt
<char_count> configures the input buffer level at which the specified interface may send the XON character (ASCII 11₁₆), assert the DTR line, and/or assert the RTS line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>char_count</i>	numeric	1 through 99 for built-in 1 through 8191 for HP E1324A	none

Comments

- To determine the size of the input buffer of the serial interface you are using, send **SYST:COMM:SER[n]:PACE:THR:STARt? MAX**. The returned value will be the buffer size less one.
- ...STARt must be set to less than ...STOP.
- The ...THR:STAR command has no effect unless ...PACE:PROT XON, ...CONT:DTR IBF, or ...CONT:DTR IBF has been sent.
- **Related Commands:** ...PACE:PROT XON | NONE, ...CONT:DTR, ...CONT:RTS
- ***RST Condition:** No change

Example

Set interface to send XON when input buffer contains 10 characters.

```
SYST:COMM:SER0:PACE:PROT XON
SYST:COMM:SER0:PACE:THR:STAR 10
```

**:COMMunicate
:SERial[n] [:RECeive]
:PACE :THReshold
:STARt?**

SYSTem:COMMunicate:SERial[n] [:RECeive]:PACE:THReshold:STARt?
 [MIN | MAX] returns:

- The current start threshold if no parameter is sent.
- The maximum allowable setting if MAX is sent.
- The minimum allowable setting if MIN is sent.

Comments

- To determine the size of the input buffer of the serial interface you are using, send **SYST:COMM:SER[n]:PACE:THR:STARt? MAX**. The returned value will be the buffer size.

Example

Return current start threshold

```
SYST:COMM:SER0:PACE:THR:STAR? query for threshold value
enter statement statement enters a numeric value
```

SYSTem:COMMunicate :SERial[n] [:RECEive] :PACE :THReshold :STOP

**:COMMunicate
:SERial[n] [:RECEive]
:PACE :THReshold
:STOP**

SYSTem:COMMunicate:SERial[n] [:RECEive]:PACE:THReshold:STOP
< *char_count* > configures the input buffer level at which the specified interface may send the XOFF character (ASCII 13₁₆), de-assert the DTR line, and/or de-assert the RTS line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>char_count</i>	numeric	1 through 99 for built-in 1 through 8191 for HP E1324A	none

Comments

- To determine the size of the input buffer of the serial interface you are using, send SYST:COMM:SER[n]:PACE:THR:STOP? MAX. The returned value will be the buffer size.
- ...STOP must be set to greater than ...START.
- The ...THR:STOP command has no effect unless ...PACE:PROT XON, ...CONT:DTR IBF, or ...CONT:DTR IBF has been sent.
- **Related Commands:** ...PACE:PROT XON | NONE, ...CONT:DTR, ...CONT:RTS
- ***RST Condition:** No change

Example

Set interface to send XOFF when input buffer contains 80 characters.

SYST:COMM:SER0:PACE:THR:STOP 80

**:COMMunicate
:SERial[n] [:RECEive]
:PACE :THReshold
:STOP?**

SYSTem:COMMunicate:SERial[n] [:RECEive]:PACE:THReshold:STOP?
[MIN | MAX] returns:

- The current stop threshold if no parameter is sent.
- The maximum allowable setting if MAX is sent.
- The minimum allowable setting if MIN is sent.

Comments

- To determine the size of the input buffer of the serial interface you are using, send SYST:COMM:SER[n]:PACE:THR:STOP? MAX. The returned value will be the buffer size.

Example

Return current stop threshold

SYST:COMM:SER0:PACE:THR:STOP? *query for threshold*
enter statement *statement enters a numeric value*

SYSTem :COMMunicate :SERial[n] [:RECEive] :PARity :CHECK

**:COMMunicate
:SERial[n] [:RECEive]
:PARity :CHECK**

SYSTem:COMMunicate:SERial[n] [:RECEive]:PARity:CHECK <check_ctrl> controls whether or not the parity bit in received serial data frames will be considered significant.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>check_ctrl</i>	boolean	0 1 OFF ON	none

Comments

- When *check_ctrl* is set to 0 or OFF, received data is not checked for correct parity. Transmitted data still includes the type of parity configured with ...PARity:TYPE.
- DIAG:BOOT:COLD will set ...CHECK to OFF.
- **Related Commands:** SYST:COMM:SER[n]:PARity:TYPE
- ***RST Condition:** No change

Example

Set parity check to ON
SYST:COMM:SER0:PAR:CHEC ON

**:COMMunicate
:SERial[n] [:RECEive]
:PARity :CHECK?**

SYSTem:COMMunicate:SERial[n] [:RECEive]:PARity:CHECK? returns the state of parity checking.

Example

Is parity checking on or off?
SYST:COMM:SER0:PAR:CHEC?
enter statement *statement enters 0 or 1*

**:COMMunicate:
SERial[n] [:RECEive]
:PARity [:TYPE]**

SYSTem:COMMunicate:SERial[n] [:RECEive]:PARity[:TYPE] <type> Configures the type of parity to be checked for received data, and generated for transmitted data.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>type</i>	discrete	EVEN ODD ZERO ONE NONE	none

Comments

- Attempting to set *type* to other than those values shown will result in an error -222.

SYSTem:COMMunicate: SERial[n] [:RECEive] :PARity [:TYPE]

- The following table defines each value of *type*:

Value	Definition
EVEN	If ...PARity:CHECK is ON, the received parity bit must maintain even parity. The transmitted parity bit will maintain even parity.
ODD	If ...PARity:CHECK is ON, the received parity bit must maintain odd parity. The transmitted parity bit will maintain odd parity.
ZERO	If ...PARity:CHECK is ON, the received parity bit must be a zero. The transmitted parity bit will be a zero.
ONE	If ...PARity:CHECK is ON, the received parity bit must be a logic one. The transmitted parity bit will be a logic one.
NONE	A parity bit must not be received in the serial data frame. No parity bit will be transmitted.

- While this command operates independently of either the ...BITS or ...SBITs commands, there are two combinations which are disallowed because of their data frame bit width. The following table shows the possible combinations:

...BITS	...PARity:TYPE	...SBITs	Frame Bits
7	NONE	1	9 - disallowed
7	NONE	2	10
7	Yes	1	10
7	Yes	2	11
8	NONE	1	10
8	NONE	2	11
8	Yes	1	11
8	Yes	2	12 - disallowed

- Received parity will not be checked unless ...PAR:CHEC ON is has been sent. Transmitted data will include the specified parity whether ...PAR:CHEC is ON or OFF.
- DIAG:BOOT:COLD will set ...PARity to NONE.
- **Related Commands:** ...PAR:CHEC 1 | 0 | ON | OFF, ...SER[n]:BITS 7 | 8, ...SER[n]:SBITs 1 | 2
- ***RST Condition:** No change

Example Set parity check/generation to ODD.

```

SYST:COMM:SER0:PAR ODD           Set parity type
SYST:COMM:SER0:PAR:CHEC ON       Enable parity check/gen.
    
```

SYSTem :COMMunicate :SERial[n] [:RECeive] :PARity [:TYPE]?

**:COMMunicate
:SERial[n] [:RECeive]
:PARity [:TYPE]?**

SYSTem:COMMunicate:SERial[n] [:RECeive]:PARity[:TYPE]? returns the type of parity checked and generated.

Example What type of parity checking is set?

SYST:COMM:SER0:PAR?
enter statement

*ask for parity type
returns the string EVEN, ODD,
ZERO, ONE, or NONE*

**:COMMunicate
:SERial[n] [:RECeive]
:SBITS**

SYSTem:COMMunicate:SERial[n] [:RECeive]:SBITS <sbits> Sets the number of stop bits to be used to transmit and receive data.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>sbits</i>	numeric	1 2 MIN MAX	none

Comments

- Attempting to set *sbits* to other than those values shown will result in an error -222.
- While this command operates independently of either the ...BITS or ...PARity:TYPE commands, there are two combinations which are disallowed because of their data frame bit width. The following table shows the possible combinations:

...BITS	...PARity:TYPE	...SBITS	Frame Bits
7	NONE	1	9 - disallowed
7	NONE	2	10
7	Yes	1	10
7	Yes	2	11
8	NONE	1	10
8	NONE	2	11
8	Yes	1	11
8	Yes	2	12 - disallowed

- DIAG:BOOT:COLD will set ...SBITS to 1.
- Related Commands: SYST:COMM:SER[n]:BAUD
- *RST Condition: No change

Example Configuring for 2 stop bits.

SYST:COMM:SER0:SBITS 2

SYSTem:COMMunicate :SERial[n] [:RECEive] :SBITs?

**:COMMunicate
:SERial[n] [:RECEive]
:SBITs?**

SYSTem:COMMunicate:SERial[n] [:RECEive]:SBITs? [MIN | MAX] returns:

- The current stop bit setting if no parameter is sent.
- The maximum allowable setting if MAX is sent.
- The minimum allowable setting if MIN is sent.

Example Querying the current stop bit configuration.

SYST:COMM:SER0:SBITs? *:REC is implied*
 enter statement *statement enters 1 or 2*

**:COMMunicate
:SERial[n] :TRANsmitt
:AUTO**

SYSTem:COMMunicate:SERial[n]:TRANsmitt:AUTO <auto_ctrl> when ON, sets the transmit pacing mode to be the same as that set for receive pacing. When OFF, the transmit pacing mode may be set independently of the receive pacing mode.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>auto_ctrl</i>	boolean	0 1 OFF ON	none

Comments

- For an HP E1324A, AUTO is always ON. Trying to set OFF or 0 will generate an error.
- DIAG:BOOT:COLD will set ...AUTO to ON.
- **Related Commands:** SYST:COMM:SER[n]:REC:PAC:PROT, SYST:COMM:SER[n]:TRAN:PAC:PROT
- ***RST Condition:** ... TRAN:AUTO ON

Example Link transmit pacing with receive pacing

SYST:COMM:SER0:TRAN:AUTO ON

**:COMMunicate
:SERial[n] :TRANsmitt
:AUTO?**

SYSTem:COMMunicate:SERial[n]:TRANsmitt:AUTO? returns the current state of receive to transmit pacing linkage.

Comments

- For an HP E1324A, AUTO is always ON. In this case ...AUTO? will always return a 1.

Example Is AUTO ON or OFF?

SYST:COMM:SER0:TRAN:AUTO?
 enter statement *statement enters the number 1 or 0*

**:COMMunicate
:SERial[n]:TRANsmit
:PACE [:PROTOcol]**

SYSTem:COMMunicate:SERial[n]:TRANsmit:PACE[:PROTOcol]
<protocol> enables or disables the transmit pacing (XON/XOFF) protocol.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>protocol</i>	discrete	XON NONE	none

Comments

- For an HP E1324A, AUTO is always ON. In this case ...TRAN:PACE will also set ...[RECeive]:PACE
- Receipt of an XOFF character (ASCII 19₁₀, 13₁₆) will hold off transmission of data until an XON character (ASCII 17₁₀, 11₁₆) is received.
- DIAG:BOOT:COLD will set ...PACE to XON.
- **Related Commands:** SYST:COMM:SER[n]:TRAN:AUTO
- ***RST Condition:** No change

Example

Set XON/XOFF transmit pacing
SYST:COMM:SER0:TRAN:PACE:PROT XON

**:COMMunicate
:SERial[n] :TRANsmit
:PACE [:PROTOcol]?**

SYSTem:COMMunicate:SERial[n]:TRANsmit:PACE[:PROTOcol]? returns the current transmit pacing protocol.

Example

Check transmit pacing protocol
SYST:COMM:SER0:TRAN:PACE:PROT?
enter statement *statement enters the string "XON" or "NONE"*

:DATE

SYSTem:DATE <year>, <month>, <day> sets the Command Module's internal calendar.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>year</i>	numeric	must round to 1980 to 2079	none
<i>month</i>	numeric	must round to 1 to 12	none
<i>day</i>	numeric	must round to 1 through last day of month	none

Comments

- The upper limit on the day parameter is dependent on the month parameter and may be dependent on the year parameter in the case of a leap year.

:TIME SYSTem:TIME <hour>, <minute>, <second> sets the Command Module's internal clock.

Parameter Name	Parameter Type	Range of Values	Default Units
<i>hour</i>	numeric	must round to 0 to 23	none
<i>minute</i>	numeric	must round to 0 to 59	none
<i>second</i>	numeric	must round to 0 to 60	none

- Comments**
- **Related Commands:** SYST:DATE, SYST:DATE?, SYST:TIME?
 - ***RST Condition:** *RST does not change the Command Module's real time clock.

Example Setting the system time

SYST:TIME 14,30,20 *set 2:30:20 PM*

- :TIME?** SYSTem:TIME? [MAX|MIN,MAX|MIN,MAX|MIN] returns:
- When no parameter is sent; the current system time in the form + HH, + MM, + SS, where HH can be 0 through 23 hours, MM can be 0 through 59 minutes, and SS can be 0 through 60 seconds.
 - When parameters are sent; the minimum or maximum allowable values for each of the three parameters. The parameter count must be three.

Example Querying the system time

SYST:TIME? *ask for current time*
input values of hour,min,sec *read back time*

:VERSion? SYSTem:VERSion? Returns the SCPI version for which this instrument complies.

- Comments**
- The returned information is in the format: YYYY.R; where YYYY is the year, and R is the revision number within that year.
 - **Related Commands:** *IDN?

Example Determine compliance version for this instrument.

SYST:VERS?
enter statement *Statement enters 1990.0*

VXI

The VXI command subsystem provides for:

- Determining the number, type, and logical address of the devices (instruments) installed in the C-size mainframe.
- Direct access to VXIbus A16 registers within devices installed in the Mainframe.
- Sending commands using the word serial protocol.
- Access to message-based devices from an RS-232 terminal.

Subsystem Syntax VXI

```
:CONFigure
:CTABLE <address >
:CTABLE?
:DCTable <address >
:DCTable?
:DeviceLADd?
:DeviceLISt?
:DeviceNUMber?
:ETABLE <address >
:ETABLE?
:HEIRarchy?
:ALL?
:INformation?
:ALL?
:ITABLE <address >
:ITABLE?
:LADDRESS?
:MEXTender?
:MEXTender
:ECLTrg <n >
:INTerrupt <n >
:TTLTrg <N >
:MTABLE <address >
:MTABLE?
:QUERY? <logical_addr >
:READ? <logical_addr >, <register_num >
:RECEive
[:MESSAge]? <logical_addr >[, <end_of_msg >]
:REGister
:READ? <register >
:WRITE? <register >, <data >
:RESet <logical_addr >
:RESet? :SEND
:COMManD <logical_addr >, <command >[, <data >]
:COMManD? <logical_addr >, <command >[, <data1 >[, <data2 >]]
:MESSAge <logical_addr >, <msg_string >[, <end_flag >]
:WRITe <logical_addr >, <register_num >, <data >
:WSProtocol
:COMManD
:AHLine <hand_id >, <line_number >
:AILine <int_id >, <line_number >
:AMControl <response_mask >
:BAVailable <end_bit >
:BNO <top_level >
:BRQ
:CEVent <enable >, <event_number >
:CLR
:CLOCK
:CRESPonse <response_mask >
:ENO
:GDEvice <logical_address >
:ICOMmander <logical_address >
:RDEvice <logical_address >
:RHANdlers
:RHLine <hand_id >
:RILine <int_id >
:RINTerrupter
:RMOdId
:RPERror
```

```

:RPRotocol
:RSTB
:RSARea
:SLOCK
:SLModid <enable>,<modid_6-0>
:SLOCK
:SUModid <enable>,<modid_12-7>
:TRIGger
:MESSage
:SEND <message_string> [[,<enable>]]
:RECeive? <count|terminator>
:QUERy
[:ANY]? <data>
:AHLine? <hand_id>,<line_number>
:AILine? <int_id>,<line_number>
:AMControl? <response_mask>
:ANO?
:BRQ?
:CEVent? <enable>,<event_number>
:CRESpouse? <response_mask>
:ENO?
:RDEvice? <logical_address>
:RHANdlers?
:RHLine? <hand_id>
:RILine? <int_id>
:RINTerrupter?
:RMOdId?
:RPERror?
:RPRotocol?
:RSTB?
:RSARea?
:SLModid? <enable>,<modid_6-0>
:SUModid? <enable>,<modid_12-7>
:RESPonse?
    
```

:CONFIgure:CTABle

VXI:CONF:CTAB <address> Links a user-defined Commander/Servant Hierarchy table to the Command Module (resource manager) processor. The Command Module must be the acting resource manager in order for the table to be implemented.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
address	numeric	(DIAG:NRAM:ADDR?)	none

Comments

- Be certain that *address* specifies the starting address of the area in User RAM (allocated using DIAG:NRAM:CREate) where you stored the Commander/Servant Hierarchy table.
- Tables must start on an even address. Note that DIAG:NRAM:CREate allocates RAM for the table with an even starting address.
- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- Setting *address* to 0 prevents the parameters defined by the table from being invoked when the system is re-booted. However, the table remains in user RAM.
- For more information see the "User Defined Commander/Servant Hierarchies" section in Chapter 2 of this manual.

VXI:CONFigure:CTABLE?

- **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:ADDR?, DIAG:DOWN, VXI:CONF:CTABLE?

Example Link a Commander/Servant Hierarchy Table to the Processor

```

DIAG:NRAM:CRE < size >           allocate space for table
                                  in user RAM
DIAG:BOOT                         re-boot system to complete
                                  allocation
DIAG:NRAM:ADDR?                   get starting address of
                                  table (RAM segment)
DIAG:DOWN < address > < data >    download data into table
VXI:CONF:CTAB < address >        link table to processor
DIAG:BOOT                         re-boot system to implement
                                  table

```

:CONFigure:CTABLE? VXI:CONF:CTAB? Returns the starting address of the user's Commander/Servant Hierarchy Table.

Example Query Address of the Commander/Servant Hierarchy Table.

```

VXI:CONF:CTABLE?                 ask for address
enter statement                   return address

```

:CONFigure:DCTable. VXI:CONF:DCT < address > Links a user-defined Dynamic Configuration table to the Command Module (resource manager) processor. The Command Module must be the acting resource manager in order for the table to be implemented.

Parameter Name	Parameter Type	Range of Values	Default Units
address	numeric	(DIAG:NRAM:ADDR?)	none

- Comments**
- Be certain that *address* specifies the starting address of the area in User RAM (allocated using DIAG:NRAM:CREate) where you stored the Dynamic Configuration Table data.
 - Tables must start on an even address. Note that DIAG:NRAM:CREate allocates RAM for the table with an even starting address.
 - *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
 - Setting *address* to 0 prevents the parameters defined by the table from being invoked when the system is re-booted. However, the table remains in user RAM.
 - For more information see the "User Defined Dynamic Configuration" section in Chapter 2 of this manual.
 - **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:ADDR?, DIAG:DOWN, VXI:CONF:DCTable?

Example Link a Dynamic Configuration Table to the Processor

DIAG:NRAM:CRE < size >	<i>allocate space for table in user RAM</i>
DIAG:BOOT	<i>re-boot system to complete the allocation</i>
DIAG:NRAM:ADDR?	<i>get starting address of table (RAM segment)</i>
DIAG:DOWN < address > < data >	<i>download data into table</i>
VXI:CONF:DCTAB < address >	<i>link table to processor</i>
DIAG:BOOT	<i>re-boot system to implement table</i>

:CONFigure:DCTable?

VXI:CONF:DCTable? Returns the starting address of the user's Dynamic Configuration Table.

Example Query Address of Dynamic Configuration table.

VXI:CONF:DCTable?	<i>ask for address</i>
enter statement	<i>return address</i>

:CONFigure :DLADdress?

VXI:CONF:DLAD? returns a comma separated decimal numeric list of device logical addresses currently installed in the mainframe. If the Command Module is not the resource manager, it only returns the logical addresses of the devices in its servant area.

Comments

- Use the VXI:CONF:DNUM? command to determine the number of values which will be returned by VXI:CONF:DLAD?.
- Use each of the logical addresses returned by VXI:CONF:DLAD? with VXI:CONF:DLIS? to determine the types of devices installed.
- VXI:CONF:DEVICELAD? is also accepted.
- This command has been retained for compatibility with existing programs. For new programs you should use the VXI:CONF:LADD? command.
- **Related Commands:** VXI:CONF:DLIS?, VXI:CONF:DNUM?, VXI:CONF:LADD?

Example Determining the device addresses within the system

VXI:CONF:DLAD?	<i>query for list of addresses.</i>
enter statement	<i>list of addresses.</i>

:CONFigure:DLIS?

VXI:CONF:DLIS? [*<logical_addr>*] returns information about the device specified by *logical_addr*. Response data is in the form:

n1, n2, n3, n4, n5, n6, c1, c2, c3, c4, c5, s1, s2, s3, s4

Where the fields above are defined as:

- n** fields Indicate numeric data response fields.
- c** fields Indicate character data response fields.
- s** fields Indicate string data response fields.

- n1** **Device's Logical Address.** A number from 0 to 255.
- n2** **Commander's Logical Address.** A number from -1 to 255; -1 means this device has no commander.
- n3** **Manufacturer's ID.** A number from 0 to 4095.
- n4** **Model Code.** A number from 0 to 65535, chosen by the manufacturer to signify the model of this device.
- n5** **Slot Number.** A number between -1 and the number of slots in this mainframe; -1 indicates that the slot associated with this device is unknown. This is always -1 for B size mainframes.
- n6** **Slot 0 Logical Address.** A number from 0 to 255.
- c1** **Device Class.** 3 data characters; EXT|HYB|MEM|MSG|REG|VME. EXT = Extended device, HYB = hybrid device (e.g. IBASIC), MEM = memory device, MSG = Message-based device, REG = Register-based device, VME = VME device
- c2** **Memory Space.** Up to 4 data characters; A16|A24|A32|NONE|RES. A16 = A16 addressing mode, A24 = A24 addressing mode, A32 = A32 addressing mode, NONE = no addressing mode, RES = reserved.
- c3** **Memory Offset.** 10 data characters which define the base address of the A24 or A32 address space on the device. This value is expressed in hex format (first two characters are #H).
- c4** **Memory Size.** 10 data characters which define the size of the A24 or A32 address space in bytes. This value is expressed in hex format (first two characters are #H).
- c5** **Pass/Failed.** Up to 5 data characters which define the status of the device; FAIL | IFAIL | PASS | READY. FAIL = failed self-test, IFAIL = configuration register initialization fails, PASS = self-test passed, READY = ready to receive commands
- s1** **Extended Field 1.** Not currently used; returns ""
- s2** **Extended Field 2.** Not currently used; returns ""
- s3** **Extended Field 3.** Not currently used; returns ""
- s4** **Manufacturer's Specific Comments.** Up to 80 character string contains manufacturer specific data in string response data format. This field is sent with a 488.2 string response data format, and will contain the instrument name and its IEEE 488.1 secondary address unless a start-up error is detected. In that case, this field will contain one or more error codes in the form "CNFG ERROR: n, m, ...,z" . See Appendix B, Table B-3 for a complete list of these codes.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	numeric	0-255 (or nothing)	none

Comments

- When *logical_addr* is not specified, VXI:CONF:DLIS? returns information for each of the devices installed, separated by semicolons. If the Command Module is not the resource manager, it returns information on only the devices in its servant area.
- Cards which are part of a combined instrument such as a switchbox or scanning voltmeter always return the same manufacturer's comments as the first card in the instrument. Information in the other fields correspond to the card for which the Logical Address was specified.
- This command has been retained for compatibility with existing programs. For new programs you should use the VXI:CONF:INF? and VXI:CONF:HEIR? commands.
- Related Commands: VXI:CONF:DLAD?, VXI:CONF:DNUM?, VXI:CONF:INF?, VXI:CONF:HEIR?

Example

Querying the device list for the System Instrument

```

dimension string[1000]           string size large in case of
                                  multiple device list
VXI:CONF:DLIS? 0                Ask for the device list for the
                                  System Instrument
enter string                      enter return data into string
    
```

Example response data (no error): +0, -1, +4095, +1301, +0, +0, HYB, NONE, #H00000000, #H00000000, READY, "", "", "", "SYSTEM INSTALLED AT SECONDARY ADDR 0"

Example response data (with error): +255, +0, +4095, +65380, -1, +0, REG, A16, #H00000000, #H00000000, READY, "", "", "", "CNFG ERROR: 11"

:CONFigure :DNUMBER?

VXI:CONF:DNUM? returns the number of devices installed in the mainframe (including the System Instrument itself). If the Command Module is not the resource manager, it returns the number of devices in its servant area.

Comments

- Use the VXI:CONF:DNUM? command to determine the number of values which will be returned by VXI:CONF:DLAD?.
- This command has been retained for compatibility with existing programs. For new programs you should use the VXI:CONF:NUMB? command.
- Related Commands: VXI:CONF:DLAD?, VXI:CONF:DLIS?

Example

Determining the number of devices within the system

```

VXI:CONF:DNUM?                  query the number of devices
enter statement                  input number of devices
    
```

**:CONFigure
:HIERarchy?**

VXI:CONF:HIER? Returns current hierarchy configuration information about the selected logical address. The individual fields of the response are comma separated. If the information about the selected logical address is not available from the destination device (i.e., the requested device is not in the mainframe or the Command Module's servant area) then Error -224 ("parameter error") will be set and no response data will be sent.

Comments

- This command returns the following values:

Logical address: an integer between -1 and 255 inclusive. -1 indicates that the device has no logical address.

Commander's logical address: an integer between -1 and 255 inclusive. -1 indicates that the device has no commander or that the commander is unknown.

Interrupt handlers: a comma separated list of seven integers between 0 and 7 inclusive. Interrupt lines 1-7 are mapped to the individual return values. 0 is used to indicate that the particular interrupt handler is not configured. A set of return values of 0,0,0,5,2,0,6 would indicate that:

- handler 4 is configured to handle interrupts on line 5
- handler 5 is configured to handle interrupts on line 2
- handler 7 is configured to handle interrupts on line 6
- handlers 1, 2, 3, and 6 are not configured

Interrupters: a comma separated list of seven integers between 0 and 7 inclusive. Interrupt lines 1-7 are mapped to the individual return values. 0 indicates that the particular interrupter is not configured. A set of return values of 0,0,0,5,2,0,6 would indicate that:

- interrupter 4 is configured to handle interrupts on line 5
- interrupter 5 is configured to handle interrupts on line 2
- interrupter 7 is configured to handle interrupts on line 6
- interrupters 1, 2, 3, and 6 are not configured

Pass/Failed: an integer which contains the pass/fail status of the specified device encoded as follows:

0 = FAIL, 1 = IFAIL, 2 = PASS, 3 = READY

Manufacturer specific comment: up to an 80 character quoted string that contains manufacturer specific data. It is sent with a 488.2 string response data format, and will contain the instrument name and its IEEE 488.1 secondary address unless a start-up error is detected. In that case, this field will contain one or more error codes in the form "CNFG ERROR: n, m, ...,z". See Appendix B, Table B-3 for a complete list of these codes.

- Cards which are part of a combined instrument such as a switchbox or scanning voltmeter always return the same manufacturer's comments as the first card in the instrument. Information in the other fields correspond to the card for which the Logical Address was specified.
- **Related Commands:** VXI:SEL, VXI:CONF:HEIR:ALL?, VXI:CONF:LADD?

**:CONFIgure
:HIERarchy:ALL?**

VXI:CONF:HIER:ALL? Returns the configuration information about all logical addresses in the mainframe, or the devices in the Command Module's servant area if the Command Module is not the resource manager. The information is returned in the order specified in the response to **VXI:CONF:LADD?**. The information about multiple logical addresses will be semicolon separated and follow the IEEE 488.2 response message format. Individual fields of the output are comma separated.

Comments

- **Related Commands:** **VXI:SEL**, **VXI:CONF:HEIR?**, **VXI:CONF:LADD?**

**:CONFIgure
:INFIguration?**

VXI:CONF:INF? Returns the static information about the selected logical address (see **VXI:SELEct**). The individual fields of the response are comma separated. If the information about the selected logical address is not available from the destination device (i.e., the requested device is not in the mainframe or the Command Module's servant area) then Error -224 ("parameter error") will be set and no response data will be sent. The command returns the following values:

- **Logical address:** an integer between -1 and 255 inclusive. -1 indicates that the device has no logical address.
- **Manufacturer ID:** an integer between -1 and 4095 inclusive. -1 indicates that the device has no Manufacturer ID.
- **Model code:** an integer between -1 and 65535 inclusive. -1 indicates that the device has no model code.
- **Device class:** an integer between 0 and 5 inclusive. 0 = VXIbus memory device, 1 = VXIbus extended device, 2 = VXIbus message based device, 3 = VXIbus register based device, 4 = Hybrid device, 5 = Non-VXIbus device
- **Address space:** an integer between 0 and 15 inclusive, which is the sum of the binary weighted codes of the address space(s) occupied by the device. 1 = The device has A16 registers, 2 = The device has A24 registers, 4 = The device has A32 registers, 8 = The device has A64 registers
- **A16 memory offset:** an integer between -1 and 65535 inclusive. Indicates the base address for any A16 registers (other than the VXIbus defined registers) which are present on the device. -1 indicates that the device has no A16 memory.
- **A24 memory offset:** an integer between -1 and 16777215 inclusive. Indicates the base address for any A24 registers which are present on the device. -1 indicates that the device has no A24 memory.
- **A32 memory offset:** an integer between -1 and 4294967295 inclusive. Indicates the base address for any A32 registers which are present on the device. -1 indicates that the device has no A32 memory.
- **A16 memory size:** an integer between -1 and 65535 inclusive. Indicates the the number of bytes reserved for any A16 registers (other than the

VXI:CONFigure :INFormation:ALL?

VXIbus defined registers) which are present on the device. -1 indicates that the device has no A16 memory.

- **A24 memory size:** an integer between -1 and 16777215 inclusive. Indicates the number of bytes reserved for any A24 registers which are present on the device. -1 indicates that the device has no A24 memory.
- **A32 memory size:** an integer between -1 and 4294967295 inclusive. Indicates the number of bytes reserved for any A32 registers which are present on the device. -1 indicates that the device has no A32 memory.
- **Slot number:** an integer between -1 and the number of slots which exist in the cage. -1 indicates that the slot which contains this device is unknown.
- **Slot 0 logical address:** an integer between -1 and 255 inclusive. -1 indicates that the Slot 0 device associated with this device is unknown.
- **Subclass:** an integer representing the contents of the subclass register. -1 indicates that the subclass register is not defined for this device.
- **Attribute:** an integer representing the contents of the attribute register. -1 indicates that the attribute register is not defined for this device.
- **Manufacturer specific comment:** up to an 80 character quoted string that contains manufacturer specific data. It is sent with a 488.2 string response data format,, and will contain the instrument name and its IEEE 488.1 secondary address unless a start-up error is detected. In that case, this field will contain one or more error codes in the form "CNFG ERROR: n, m, ...,z". See Appendix B, Table B-3 for a complete list of these codes.

Comments

- **Related Commands:** VXI:SEL, VXI:CONF:INF:ALL?, VXI:CONF:LADD?

Example

Get static information on the currently selected logical address.

VXI:SEL 0	<i>select the logical address</i>
VXI:CONF:INF?	<i>ask for data</i>
enter statement	<i>return data</i>

:CONFigure :INFormation:ALL?

VXI:CONF:INF:ALL? Returns the static information about all logical addresses. The information is returned in the order specified in the response to VXI:CONF:LADD?. The information about multiple logical addresses will be semicolon separated and follow the IEEE 488.2 response message format. Individual fields of the output are comma separated.

Comments

- **Related Commands:** VXI:SEL, VXI:CONF:INF?, VXI:CONF:LADD?

:CONFigure:ITABle

VXI:CONF:ITAB <address> Links a user-defined Interrupt Line Allocation table to the Command Module (resource manager) processor. The Command Module must be the acting resource manager in order for the table to be implemented.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
address	numeric	(DIAG:NRAM:ADDR?)	none

Comments

- Be certain that *address* specifies the starting address of the area in User RAM (allocated using DIAG:NRAM:CREate) where you stored the Interrupt Line Allocation Table data.
- Tables must start on an even address. Note that DIAG:NRAM:CREate allocates RAM for the table with an even starting address.
- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- Setting *address* to 0 prevents the parameters defined by the table from being invoked when the system is re-booted. However, the table remains in user RAM.
- For more information see the "User Defined Interrupt Line Allocation" section in Chapter 2 of this manual.
- **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:ADDR?, DIAG:DOWN, VXI:CONF:ITABle?

Example Link an Interrupt Line Allocation Table to the Processor

```

DIAG:NRAM:CRE < size >           allocate space for table
                                  in user RAM
DIAG:BOOT                          re-boot system to complete
                                  the allocation
DIAG:NRAM:ADDR?                    get starting address of
                                  table (RAM segment)
DIAG:DOWN < address > < data >    download data into table
VXI:CONF:ITAB < address >         link table to processor
DIAG:BOOT                          re-boot system to implement
    
```

:CONFigure:ITABle?

VXI:CONF:ITAB? Returns the starting address of the user's Interrupt Line Allocation Table.

Example Query Address of Interrupt Line Allocation table.

```

VXI:CONF:ITABle?                 ask for address
enter statement                   return address
    
```

VXI:CONFigure :LADdress?

**:CONFigure
:LADdress?**

VXI:CONF:LADD? Returns a comma separated list of logical addresses of devices in the mainframe, or a list of devices in the Command Module's servant area if the Command Module is not the resource manager. This is an integer between 1 and 256 inclusive. The logical address of the device responding to the command will be the first entry in the list. If the command is received by a device other than the resource manager, the response will contain the logical address of the destination device followed by a list of devices which are immediate servants to the destination device.

Comments

- **Related Commands:** VXI:SEL, VXI:CONF:NUMB?

**:CONFigure:LADdress
:MEXTender?**

VXI:CONF:LADD:MEXT? Returns a comma separated list of logical addresses of mainframe extender devices in the system. This is an integer between 1 and 256 inclusive. If there are no extender devices in the system a -1 will be returned. An error is reported if the command is received by a device other than the resource manager.

Comments

- **Related Commands:** VXI:SEL, VXI:CONF:NUMB:MEXT?

**:CONFigure:MEXTender
:ECLTrg**

VXI:CONF:MEXT:ECLT<n> <direction> is used configure the selected mainframe extender to direct the ECL trigger specified by <n>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>direction</i>	discreet	IN OUT NONE	none

Comments

- Select the logical address of the extender to access with the VXI:SElect.
- The trigger line affected is specified in the ECLTrg node of the command by an integer ranging from 0 to 5. Integers greater than 5 will generate the error "-113 Undefined header".
- A mainframe extender can direct a trigger line into or out of the VXIbus card cage (mainframe) that it is plugged into.
- If you specify NONE the trigger line will be disabled and will not be directed in or out.
- Some mainframe extender devices do not support some trigger lines. These commands will determine whether the specified trigger line is supported before it attempts to execute the command. If the trigger line is not supported a trigger not supported error will be returned.
- This command can only be executed by the System Instrument in a command module that is serving as resource manage for the entire VXIbus system.
- **Related Commands:** VXI:CONF:MEXT:INT, VXI:CONF:MEXT:TTLT, VXI:ROUT:ECLT

VXI :CONFigure:MEXTender :INTerrupt < n >

Example Direct ECL trigger line 1 from a card cage with "child side" extender at logical address 5 to an extended card cage with a "parent side extender" of logical address 6.

```
VXI:SEL 5                               Select logical address 5
VXI:CONF:MEXT:ECLT1 OUT                 Configure the logical address 5
                                         extender as OUT.

VXI:SEL 6                               Select logical address 6
VXI:CONF:MEXT:ECLT1 IN                  Configure the logical address 6
                                         extender as IN.
```

**:CONFigure:MEXTender
:INTerrupt < n >**

Parameters

VXI:CONF:MEXT:INT < n > < direction > is used to configure the selected mainframe extender to direct the interrupt line specified by < n > .

Parameter Name	Parameter Type	Range of Values	Default Units
<i>direction</i>	discreet	IN OUT NONE	none

Comments

- Select the logical address of the extender to access is with VXI:SElect.
- The interrupt line affected is specified in the INTerrupt node of the command by a number ranging from 1 to 7. Numbers greater than 7 and less than 1 will generate the error "-113 Undefined header".
- A mainframe extender can direct an interrupt line into the VXIbus card cage (mainframe) that it is plugged into or in can direct the interrupt line out of the card cage.
- If you specify NONE the interrupt line will be disabled and will not be directed in or out.
- Some mainframe extender devices do not support directing interrupt lines. These commands will determine whether the specified interrupt line is supported before it attempts to execute the command. If the interrupt line is not supported a trigger not supported error will be returned.
- This command can only be executed by the System Instrument in a command module that is serving as resource manage for the entire VXIbus system.
- **Related Commands:** VXI:CONF:MEXT:ECLT, VXI:CONF:MEXT:TTLT, VXI:ROUT:INT

VXI:CONFigure:MEXTender :TTLTrg

Example Direct interrupt line 1 from a card cage with "child side" extender at logical address 5 to an extended card cage with a "parent side extender" of logical address 6.

```
VXI:SEL 5                               Select logical address 5
VXI:CONF:MEXT:INT1 OUT                  Configure the logical address 5
                                         extender as OUT.

VXI:SEL 6                               Select logical address 6
VXI:CONF:MEXT:INT1 IN                   Configure the logical address 6
                                         extender as IN.
```

:CONFigure:MEXTender :TTLTrg

VXI:CONF:MEXT:TTLT <direction> is used configure the selected mainframe extender to direct the TTL trigger specified by <n>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>direction</i>	discreet	IN OUT NONE	none

Comments

- Select the logical address of the extender to access with VXI:SElect.
- The trigger line affected is specified in the TTLTrg node of the command by a number ranging from 0 to 7. Numbers greater than 7 will generate the error "-113 Undefined header".
- A mainframe extender can direct a trigger line into the VXIbus card cage (mainframe) that it is plugged into or it can direct the trigger line out of the card cage.
- If you specify NONE the trigger line will be disabled and will not be directed in or out.
- Some mainframe extender devices do not support some trigger lines. These commands will determine whether the specified trigger line is supported before it attempts to execute the command. If the trigger line is not supported a trigger not supported error will be returned.
- This command can only be executed by the System Instrument in a command module that is serving as resource manage for the entire VXIbus system.
- **Related Commands:** VXI:CONF:NUMB:MEXT:INT, VXI:CONF:NUMB:MEXT:ECL, VXI:ROUT:TTLT

Example Direct TTL trigger line 1 from a card cage with "child side" extender at logical address 5 to an extended card cage with a "parent side extender" of logical address 6.

```
VXI:SEL 5                               Select logical address 5
VXI:CONF:MEXT:TTLT1 OUT                  Configure the logical address 5
                                         extender as OUT.

VXI:SEL 6                               Select logical address 6
VXI:CONF:MEXT:TTLT1 IN                   Configure the logical address 6
                                         extender as IN.
```

:CONFigure:MTABLE

VXI:CONF:MTAB <address> Links a user-defined A24/A32 Address Allocation table to the Command Module (resource manager) processor. The Command Module must be the acting resource manager in order for the table to be implemented.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
address	numeric	(DIAG:NRAM:ADDR?)	none

Comments

- Be certain that *address* specifies the starting address of the area in User RAM (allocated using DIAG:NRAM:CREate) where you stored the A24/A32 Address Allocation Table data.
- Tables must start on an even address. Note that DIAG:NRAM:CREate allocates RAM for the table with an even starting address.
- *Address* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- Setting *address* to 0 prevents the parameters defined by the table from being invoked when the system is re-booted. However, the table remains in user RAM.
- For more information see the "Reserving A24/A32 Address Space" section in Chapter 2 of this manual.
- **Related Commands:** DIAG:NRAM:CREate, DIAG:NRAM:ADDR?, DIAG:DOWN, VXI:CONF:MTABLE?

Example

Link an A24/A32 Address Allocation Table to the Processor

```

DIAG:NRAM:CRE < size >           allocate space for table
                                  in user RAM
DIAG:BOOT                          re-boot system to complete
                                  the allocation
DIAG:NRAM:ADDR?                    get starting address of
                                  table (RAM segment)
DIAG:DOWN < address > < data >    download data into table
VXI:CONF:MTAB < address >         link table to processor
DIAG:BOOT                          re-boot system to implement
                                  table
    
```

:CONFigure:MTABLE?

VXI:CONF:MTAB? Returns the starting address of the user's A24/A32 Address Allocation Table.

Example

Query Address of A24/A32 Address Allocation table.

```

VXI:CONF:MTABLE?                ask for address
enter statement                  return address
    
```

VXI:CONFigure :NUMBer?

:CONFigure :NUMBer?

VXI:CONF:LADD? Returns the number of devices in the system when it is issued to a resource manager. This is an integer between 1 and 256 inclusive. If the command is received by a device that is not the resource manager, it returns the number of devices which are immediate servants to the destination device, including the destination device. For example, a commander with 3 servants would return a value of 4, or a resource manager for a system of 4 devices would return a value of 5.

Comments

- Related Commands: VXI:SEL, VXI:CONF:LADD?

:CONFigure :NUMBer
:MEXTender?

VXI:CONF:NUMB:MEXT? Returns the number of devices in the system when it is issued to a resource manager. This is an integer between 1 and 256 inclusive, which indicates the number of mainframe extender devices in the system. If the command is received by a device other than the resource manager an error is reported.

Comments

- Related Commands: VXI:SEL, VXI:CONF:LADD?, VXI:CONF:NUMB?

:QUERy?

VXI:QUERy? <logical_addr> returns one 16 bit data word from the Data Low register of the message based device at *logical_addr*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	numeric	Must round to 0 through 255	none

Comments

- Send a Device Clear to "unlock" the System Instrument in case the device at *logical_addr* does not respond.
- VXI:QUERy? can be used to read the response in the Data Low register when the VXI:SEND:COMM command is ANY, and the command sent is a query.
- This command has been retained for compatibility with existing programs. For new programs you should use VXI:WSP:RESP?
- Related Commands: VXI:SEND:COMM, VXI:WSP:RESP?

Example

Reading the Data Low register of device at Logical Address 72.

VXI:QUERy? 72

query value of Data Low register

enter statement

input 16 bit value

:READ? **VXI:READ?** <*logical_addr*>, <*register_addr*> allows access to the entire 64 byte A16 register address space for the device specified by *logical_addr*. Since the VXIbus system is byte-addressed, while the registers are 16 bits wide, registers are specified by even addresses only. This method of identifying registers follows the VXIbus standard format.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	decimal numeric	must round to 0 through 255	none
<i>register_addr</i>	numeric	must round to an even value from 0 through 62 (3E _h)	none

Comments

- Specifying an odd register address will cause an error 2003, "Invalid word address".
- Specifying a logical address not currently in the system will cause an error 2005, "No card at logical address".
- If the Command Module is the resource manager it can read from any device within the mainframe. If the Command Module is not the Resource Manager it can only read from devices within its servant area.
- *Logical_addr* must be specified in decimal. *Register_addr* may be specified in decimal, hex (#H), octal (#Q), or binary (#B).
- Accesses are 16-bit non-privileged data accesses.
- This command has been retained for compatibility with existing programs. For new programs you should use VXI:REG:READ?
- **Related Commands:** VXI:WRITE, VXI:REG:READ?

Example Read from one of a device's configuration registers

VXI:READ? 8,0

*read ID register on device at
Logical Address 8*

enter statement

enter value from device register

VXI:RECeive[:MESSAge]?

:RECeive[:MESSAge]?

VXI:RECeive:MESSAge? <logical_addr> [, <end_of_msg>] receives a message from the message based device at *logical_addr*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	numeric	must round to 0 through 255	none
<i>end_of_msg</i>	discrete/ numeric	END LF CRLF < count >	none

Comments

- A message ends when the condition specified by the *end_of_msg* parameter is met. When *end_of_msg* specifies a count, it can range from 1 through 2,147,483,647.
- The default *end_of_msg* parameter is END.
- VXI:REC? together with VXI:SEND can be used to communicate with message-based devices from an RS-232 monitor via the Command Module. If the Command Module is the resource manager, the message-based devices can be inside or outside its servant area. If the Command Module is not the resource manager, the message-based devices must be in the Command Module's servant area.
- VXI:REC? uses the Byte Transfer Protocol which uses the DIR and DOR bits in the Response register. This protocol and DIR/DOR are described in the VXIbus System Specification Manual Revision 1.3 (HP Part Number E1400-90006).
- Send a Device Clear to "unlock" the System Instrument in case the device at *logical_addr* does not satisfy the *end_of_msg* condition (insufficient data for count, or no END | LF | CRLF).
- This command has been retained for compatibility with existing programs. For new programs you should use VXI:WSP:MESS:REC?
- **Related Commands:** VXI:SEND:MESS, VXI:WSP:MESS:REC?, VXI:WSP:MESS:SEND

Example Query for message from module at logical address 16.

```
VXI:SEND 16,"*IDN?"           send command to device at
                               logical address 16
VXI:REC? 16                   enter message
```

:REGister:READ? VXI:REG:READ? <register> returns the contents of the specified 16 bit register at the selected logical address as an integer (see VXI:SElect).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>register</i>	numeric	even numbers from 0 to 62 or register name (see below)	none

Comments

- The register parameter can be all even numbers from 0 to 62 inclusive (as a < numeric_value >) or the following (optional) words:

A16 Window: A16 Window Map Register (12)

A24Low: A24 Pointer Low register (18)

A24High: A24 Pointer High register (16)

A24 Window: A24 Window Map Register (14)

A32Low: A32 Pointer Low register (22)

A32High: A32 Pointer High register (20)

A32 Window: A32 Window Map Register (16)

ATTRibute: Attribute register (8)

DHIGH: Data High register (12)

DLOW: Data Low register (14)

DTYPe: Device Type register (2)

ETConfigure: ECL Trigger Configuration Register (22)

ICNF: Interrupt Configuration Register (18)

ICONtrol: Interrupt control register (28)

ID: ID register (0)

IStatus: Interrupt Status register (26)

LAWindow: Logical Address Configuration Register (10)

TTConfigure: TTL Trigger Configuration Register (20)

MODId: MODID register (8)

OFFSet: Offset register (6)

PROTOcol: Protocol register (8)

RESPonse: Response register (10)

SNHigh: Serial Number High register (10)

SNLow: Serial Number Low register (12)

STATus: Status register (4)

SUBClass: Subclass register (30)

UCONfigure: Utility configuration Register (24)

VNUMber: Version Number register (14)

NOTE

The optional register names are decoded into the equivalent register address. You will get correct results if you use any one of the words for a given register address, even if the word itself does not make sense for the device you are using.

- **Related Commands:** VXI:SEL, VXI:REG:WRIT

Example Read from a register on the currently selected device

VXI:READ? CONT

Read from the control register of the currently selected device

:REGister:WRITE VXI:REG:WRIT <register> , <data> writes to the specified 16 bit register at the selected logical address (see VXI:SElect).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>register</i>	numeric	even numbers from 0 to 62 or register name (see below)	none
<i>data</i>	numeric	-32768 to 32767	none

Comments

- The register parameter can be all even numbers from 0 to 62 inclusive (as a <numeric_value>) or the following (optional) words:
A16 Window: A16 Window Map Register (12)
A24 Window: A24 Window Map Register (14)
A32 Window: A32 Window Map Register (16)
CONTROL: Control Register (4)
DEXTended: Data Extended register (10)
DHIGH: Data High register (12)
DLOW: Data Low register (14)
ETConfigure: ECL Trigger Configuration Register (22)
ICNF: Interrupt Configuration Register (18)
ICONtrol: Interrupt Control register (28)
LAWindow: Logical Address Configuration Register (10)
MODid: MODID register (8)
LADDRESS: Logical Address register (0)
OFFSet: Offset register (6)
SIGNal: Signal register (8)
TTConfigure: TTL Trigger Configuration Register (20)
UCONfigure: Utility configuration Register (24)

NOTE

The optional register names are decoded into the equivalent register address. You will get correct results if you use any one of the words for a given register address, even if the word itself does not make sense for the device you are using.

- **Related Commands:** VXI:SEL, VXI:REG:READ?

Example

Write to a register on the currently selected device

VXI:REG:WRIT DHIG,64 *writes "64" to Data High register*

:RESet VXI:RESet <logical_addr> performs a Soft Reset of the device at *logical_addr*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	numeric	Must round to 0 through 255	none

Comments

- RESet sets the Sysfail Inhibit bit in the device's control register, then sets the Reset bit, waits 100 μ s, then clears Reset. When the device has passed its self test, Sysfail Inhibit is cleared. If the device fails during the reset (does not assert "Passes" within 4.9 sec), Sysfail Inhibit remains asserted.
- If the Command Module is the resource manager, it can reset any device within the mainframe. If the Command Module is not the resource manager, it can only reset devices within its servant area. You cannot use VXI:RESet to reset the Command Module (use DIAG:BOOT).
- When a device is reset, the Command Module (system instrument) will write 1's the Device Dependent bits in the device's control register.
- This command has been retained for compatibility with existing programs. For new programs you should use VXI:RESET?

Example

Reset a VXIbus device

VXI:RES 64

reset device at logical addr 64

:RESet?

VXI:RESet? resets the selected logical address. SYSFAIL generation is inhibited while the device is in the self test state. The command waits for 5 seconds or until the selected device has indicated passed (whichever occurs first). If the device passes its self test, the SYSFAIL generation is re-enabled. If the device fails the self test, then SYSFAIL generation will remain inhibited.

Comments

- The return value from this command is the state of the selected device after it has been reset. The command returns a <NR1> encoded as follows:
 - 0 = FAIL
 - 2 = PASS
 - 3 = READY
- The state of the A24/A32 enable bit is not altered by this command
- If the Command Module is the resource manager, it can reset any device within the mainframe. If the Command Module is not the resource manager, it can only reset devices within its servant area. You cannot use VXI:RESet? to reset the Command Module (use DIAG:BOOT).
- Related Commands: VXI:SEL

:ROUTE:ECLTrg VXI:ROUTE:ECLTrg < n > sets the direction for the ECL trigger line specified by < n > of all mainframe extenders in the in the system.

Comments

- The direction is set so the device selected by the VXI:SEL command can source the trigger line and all other devices in the system may monitor that trigger line.
- Some mainframe extender devices do not support some trigger lines. This command will determine whether the specified trigger line is supported while it attempts to execute the command and return a trigger not supported error if it encounters any extenders that do not support the specified trigger. It will attempt to direct all extenders that do support the specified trigger, even if it encounters some extenders that do not.
- This command can only be executed by the System Instrument in a command module that is serving as resource manager for the entire VXIbus system.
- **Related Commands:** VXI:SEL, VXI:ROUT:TTL, VXI:ROUT:INT, VXI:CONF:MEXT. . .

:ROUTE:INTerrupt VXI:ROUTE:INTerrupt < n > sets the direction for the interrupt line specified by < n > of all mainframe extenders in the system.

Comments

- The direction is set so the device selected by the VXI:SEL command can handle the interrupt line and all other devices in the system may assert that interrupt line.
- Some mainframe extender devices do not support directing interrupt lines. This command will determine whether the specified interrupt line is supported while it attempts to execute the command and return a trigger not supported error if it encounters any extenders that do not support the specified line. It will attempt to direct all extenders that do support the specified line, even if it encounters some extenders that do not.
- This command can only be executed by the System Instrument in a command module that is serving as resource manager for the entire VXIbus system.
- **Related Commands:** VXI:SEL, VXI:ROUT:TTL, VXI:ROUT:ECL, VXI:CONF:MEXT. . .

:ROUTE:TTLTrg

VXI:ROUTE:TTLTrg < n > sets the direction for the TTL trigger line specified by < n > of all mainframe extenders in the system.

Comments

- The direction is set so the device selected by the VXI:SEL command can source the trigger line and all other devices in the system may monitor that trigger line.
- Some mainframe extender devices do not support some trigger lines. This command will determine whether the specified trigger line is supported while it attempts to execute the command and return a trigger not supported error if it encounters any extenders that do not support the specified trigger. It will attempt to direct all extenders that do support the specified trigger, even if it encounters some extenders that do not.
- This command can only be executed by the System Instrument in a command module that is serving as resource manager for the entire VXIbus system.
- **Related Commands:** VXI:SEL, VXI:ROUT:INT, VXI:ROUT:ECL, VXI:CONF:MEXT...

:SElect

VXI:SElect < logical_addr > specifies the logical address which is to be used by many subsequent commands in the VXI subsystem.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	numeric	0 through 255	none

Comments

- The *RST default value for *logical_addr* is that no logical address is selected (i.e., -1). All other commands which require a logical address to be selected will respond with Error -221 ("settings conflict") if no logical address is selected.
- When a command encounters an Error -240 ("Hardware Error") the equivalent of a *RST is executed. This will cause the selected logical address to be set to -1.
- **Related Commands:** VXI:CONF:LADD?

Example

Select a logical address

VXI:SEL 64

sets the logical address to be used by subsequent VXI subsystem commands to 64.

:SElect?

VXI:SElect? returns the logical address which will be used by many subsequent commands in the VXI subsystem. If no logical address has been selected, this query will return -1.

:SEND:COMMand

VXI:SEND:COMMand <logical_addr>, <command> [<data>] sends the specified word serial *command* (and optional *data*) to *logical_addr*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	numeric	Must round to 0 through 255	none

The *command* field and any required *data* fields are specified in the table below.

Command	Data	Description
BAVailable	<byte> (0-511)	Byte Available (bit 8 = 1 = END, bits 7-0 = data byte)
CLEAr		Clear
CLOCK		Clear Lock
GDEvice	<device_addr> (0-255)	Grant Device
ICOMmander	<cmdr_addr> (0-255)	Identify Commander
SLOCK		Set Lock
TRIGger		Trigger
ANY	<cmd_word>	Specify any word serial command as a 16 bit value in cmd_word. Read response from the Data Low register using VXI:QUER?.

Comments

- *data* may be specified in decimal, hex (#H), octal (#Q), or binary (#B).
- VXI:SEND:COMMand uses the Word Serial Transfer Protocol. This protocol is described in the VXIbus System Specification Manual Revision 1.3 (HP Part Number E1400-90006). The word serial commands listed in Table 6-1 are also covered in the VXIbus System Specification Manual.
- VXI:SEND:COMMand is recommended for use with devices conforming to VXIbus REV 1.3.
- This command has been retained for compatibility with existing programs. For new programs you should use VXI:WSP:COMM.
- **Related Commands:** VXI:SEND:COMM?, VXI:WSP:COMM, VXI:WSP:QUER?

Example Send 1 data byte to logical address 241

VXI:SEND:COMM 241,BAV,452

end bit = 1 and data byte is 196

:SEND:COMMANd? VXI:SEND:COMMANd? <logical_addr>, <command> [, <data1> [, <data2>]] sends the specified word serial *command* (and optional *dataN* values) using the word-serial protocol, to the module at *logical_addr*. It then waits for and returns a 16 bit response value.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	decimal numeric	Must round to 0 through 255	none

The *command* field and any required *data* fields are specified in the following table.

Command	data1	data2	Description
AHLine	<hand_id> (1-7)	<line #> (0-7)	Assign Handler Line. A line number of 0 means the handler is to be disconnected.
ALLine	<int_id> (1-7)	<line #> (0-7)	Assign Interrupter Line. A line number of 0 means the handler is to be disconnected.
AMControl	<rpsn_mask> (0-15)		Asynchronous Mode Control
ANOperation			Abort Normal Operation
BNOOperation	<top level> (0 non-zero)		Begin Normal Operation
BREQuest	<enable> (0 1 OFF ON)	<event #> (0-127)	Byte Request Control Event
CREsponse	<rpsn_mask> (0-127)		Control Response
ENOperation			End Normal Operation
RDEvice	<logical_addr> (0-255)		Release Device
RHANDlers			Read Handlers
RHLine	<hand_id> (1-7)		Read Handler Line
RILine	<int_id> (1-7)		Read Interrupter Line
RINTerrupter			Read Interrupters
RMODid			Read MODID
RPERror			Read Protocol Error
RPRotocol			Read Protocol
RSAREa			Read servant area
RSTB			Read STB
RSAREa			Read Servant Area
SLModid	<enable> (0 1 OFF ON)	<modid> (0-127)	Set Lower MODID (lines 0-6)
SUModid	<enable> (0 1 OFF ON)	<modid> (0-63)	Set Upper MODID (lines 7-12)
ANY	<cmd_word> (-32768-32767)		Specify any VXIbus command

Comments

- *data1* and *data2* may be specified in decimal, hex (#H), octal (#O), or binary (#B) formats.
- VXI:SEND:COMMANd uses the Word Serial Transfer Protocol. This protocol is described in the VXIbus System Specification Manual Revision 1.3 (HP Part Number E1400-90006). The word serial commands

VXI:SEND[:MESSAge]

listed in Table 5-2 are also covered in the VXIbus System Specification Manual.

- VXI:SEND:COMMand? is recommended for use with devices conforming to VXIbus REV 1.3.
- This command has been retained for compatibility with existing programs. For use with new programs you should use VXI:WSP:QUER?
- **Related Commands:** VXI:SEND:COMM, VXI:WSP:QUER?

Example Read which IRQ line is used by interrupt handler in logical address 241

VXI:SEND:COMM? 241,RHLINE,2 *which line used by second handler in servant at 241*
 enter statement *return the number of the interrupt line*

:SEND[:MESSAge]

VXI:SEND:MESSAge <logical_addr>,"<msg_string>"[<end_flag>] sends the specified message string to the message based module at logical_addr.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	decimal numeric	Must round to 0 through 255	none
<i>msg_string</i>	string	ASCII chars (no nulls)	none
<i>end_flag</i>	discrete	END NOEND	none

Comments

- VXI:REC? together with VXI:SEND can be used to communicate with message-based devices from an RS-232 monitor via the Command Module. If the Command Module is the resource manager, the message-based devices can be inside or outside its servant area. If the Command Module is not the resource manager, the message-based devices must be in the Command Module's servant area.
- VXI:SEND uses the Byte Transfer Protocol which uses the DIR and DOR bits in the Response register. This protocol and DIR/DOR are described in the VXIbus System Specification Manual Revision 1.3 (HP Part Number E1400-90006).
- The last byte of *msg_string* is sent with the END bit set unless *end_flag* is specified as NOEND.
- If CR or CRLF is to be sent, they must be included in *msg_string*.
- Null characters (ascii value 0) must not occur in *msg_string*.
- This command has been retained for compatibility with existing programs. For use with new programs you should use VXI:WSP:MESS:SEND
- **Related Commands:** VXI:REC:MESS?, VXI:WSP:MESS:SEND, VXI:WSP:MESS:REC?

Example Send a message to a message-based device at logical address 16.

VXI:SEND 16,"MEAS:VOLT:DC?" *send command to message-based multimeter (last by is sent with END bit set)*

VXI:REC? 16 *retrieve voltage measurement*

:WRITE VXI:WRITE *<logical_addr>*, *<register_addr>*, *<data>* allows access to the entire 64 byte A16 register address space for the device specified by *logical_addr*. Since the VXIbus system is byte-addressed, while the registers are 16 bits wide, registers are specified by even addresses only. This method of identifying registers follows the VXIbus standard format.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>logical_addr</i>	decimal numeric	Must round to 0 through 255	none
<i>register_addr</i>	numeric	must round to an even value from 0 through 62 (3E _h)	none
<i>data</i>	numeric	must round to -32768 to 32767 (0 to FFFF _h)	none

Comments

- Specifying an odd register address will cause an error 2003,"Invalid word address".
- Specifying a logical address not currently in use in the system will cause an error 2005,"No card at logical address".
- If the Command Module is the resource manager, it can write to any device within the mainframe. If the Command Module is not the Resource Manager, it can only write to those devices within its servant area.
- *Logical_addr* must be specified in decimal. *Register_addr* and *data* may be specified in decimal, hex (#H), octal (#Q), or binary (#B).
- This command has been retained for compatibility with existing programs. For new programs you should use the VXI:REG:WRIT command.
- Accesses are 16-bit non-privileged data accesses.
- Related Commands: VXI:READ?, VXI:REG:WRIT

Write a value into a device's device dependent register.

VXI:WRIT 8,24,#H4200 *write hex 4200 (16,896 decimal) to register 24 of device at Logical Address 8*

:WSProtocol:COMMand
:command

VXI:WSProtocol:COMMand:command is a series of commands which sends the specified Word Serial Command to the address set using the VXI:SEL command and continues without waiting for a response. The response to this command can be read with the VXI:WSProtocol:RESPonse? command. The following table lists the available commands and their parameters (if any).

Command	parameter1	parameter2	Description
:AHLLine	<hand_id> (1-7)	<line_#> (0-7)	Assign Handler Line. A line number of 0 means the handler is to be disconnected.
:AILLine	<int_id> (1-7)	<line_#> (0-7)	Assign Interrupter Line. A line number of 0 means the interrupter is to be disconnected.
:AMControl	<rpn_mask> (0-15)		Asynchronous Mode Control
:ANOPeration			Abort Normal Operation
:ANY	<cmd_word> (-32768-32767)		Specify any word serial command as a 16 bit value in cmd_word.
:BAVailable	<end_bit> (0 1 OFF ON)	<byte> (0-255)	Byte Available (bit 8 = 1 = END, bits 7-0 = data byte)
:BNOPeration	<top_level> (0 1 OFF ON)		Begin Normal Operation
:BREQuest			Byte Request
:CEVent	<enable> (0 1 OFF ON)	<event_#> (0-127)	Control Event
:CLEar			Clear
:CLOCK			Clear Lock
:CRESPonse	<rspns_mask> (0-127)		Control Response
:ENOPeration			End Normal Operation
:GDEvice	<cmdr_addr> (0-255)		Grant Device
:ICOMmander			Identify Commander
:RDEvice	<logical_addr> (0-255)		Release Device
:RHANDlers			Read Handlers
:RHLine	<hand_id> (1-7)		Read Handler Line
:RILine	<int_id> (1-7)		Read Interrupter Line
:RINTerrupter			Read Interrupters
:RMODid			Read MODID
:RPERror			Read Protocol Error
:RPRotocol			Read Protocol
:RSTB			Read STB
:RSARea			Read Servant Area
:SLModid	<enable> (0 1 OFF ON)	<modid> (0-127)	Set Lower MODID (lines 0-6)
:SLOCK			Set Lock
:SUModid	<enable> (0 1 OFF ON)	<modid> (0-63)	Set Upper MODID (lines 7-12)
:TRIGger			Trigger

Comments

- *byte*, *cmd_word*, *event_number*, *hand_id*, *int_id*, *line_number*, *logical_address*, *modid*, and *response_mask* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- *end_bit* selects whether the END bit is set in the command.
- *top_level* selects whether the Top_level bit is set in the command.
- *enable* selects whether the Enable bit is set in the command.
- **Related Commands:** VXI:SEL, VXI:WSP:RESP?, VXI:WSP:QUER?

**:WSProtocol
:MESSAge:SEND**

VXI:WSP:MESS:SEND < *message string* > [, (END|NEN)] sends the specified message string to the selected logical address. The string is sent using the word serial protocol with the byte transfer protocol.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>message_string</i>	text string	Any valid text string	none
end bit	boolean	END NEN	END

Comments

- The last byte of the string is sent with the End bit set unless you specify NEN (NoEND).
- **Related Commands:** VXI:SEL, VXI:WSP:MESS:REC

**:WSProtocol
:MESSAge:RECeive?**

VXI:WSProtocol:MESSAge:RECeive? < *count* | *terminator* > receives a message from the selected logical address using both the word serial protocol and the byte transfer protocol.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<i>terminator</i>	numeric	count LF CRLF END	END

Comments

- The command will always terminate on the End bit being set. Additional termination options are on a specified number of bytes (*count*), or on a match to a particular terminator (That is, LF, CRLF, END).
- The response is returned as a string.
- **Related Commands:** VXI:SEL, VXI:WSP:MESS:SEND

**:WSProtocol:QUERY
:command?**

VXI:WSProtocol:QUERY:command? is a series of commands which sends the specified Word Serial Command to the address set using the VXI:SEL command and waits for a response. The returned value is the response to the command and is an integer. The following table lists the available commands and their parameters (if any).

Command	parameter1	parameter2	Description
:AHLLine?	< hand_id > (1-7)	< line_# > (0-7)	Assign Handler Line. A line number of 0 means the interrupter is to be disconnected.
:AILLine?	< int_id > (1-7)	< line_# > (0-7)	Assign Interrupter Line. A line number of 0 means the handler is to be disconnected.
:AMControl?	< rpn_mask > (0-15)		Asynchronous Mode Control
:ANOPeration?			Abort Normal Operation
:ANY ?	< cmd_word > (-32768-32767)		Specify any word serial command as a 16 bit value in cmd_word.
:BNOPeration?	< top_level > (0 1 OFF ON)		Begin Normal Operation
:BREQuest?		< event_# > (0-127)	Byte Request
:CEVEnt?	< enable > (0 1 OFF ON)		Control Event
:CRESPonse?	< rspns_mask > (0-127)		Control Response
:ENOPeration?			End Normal Operation
:RDEVice?	< logical_addr > (0-255)		Release Device
:RHANdlers?			Read Handlers
:RHLine?	< hand_id > (1-7)		Read Handler Line
:RILine?	< int_id > (1-7)		Read Interrupter Line
:RINTerrupter?			Read Interrupters
:RMODid?			Read MODID
:RPERror?			Read Protocol Error
:RPRotocol?			Read Protocol
:RSTB?			Read STB
:RSARea?			Read Servant Area
:SLModid?	< enable > (0 1 OFF ON)	< modid > (0-127)	Set Lower MODID (lines 0-6)
:SUModid?	< enable > (0 1 OFF ON)	< modid > (0-63)	Set Upper MODID (lines 7-12)

Comments

- *event_number, hand_id, int_id, line_number, modid, and response_mask* may be specified in decimal, hex (#H), octal (#Q), or binary (#B) formats.
- *top_level* selects whether the END bit is set in the command.
- *enable* selects whether the Enable bit is set in the command.
- **Related Commands:** VXI:SEL, VXI:WSP:COMM

**:WSProtocol
:RESPonse?**

VXI:WSProtocol:RESPonse? returns one word of data from the data low register on the selected logical address. This command obeys the Byte transfer protocol. The data is returned as an integer.

Common Command Reference

This section describes the IEEE-488.2 Common Commands that can be used to program instruments in the mainframe. Commands are listed alphabetically (Table 6-3 shows the Common Commands listed by functional group). Examples are shown when the command has parameters or returns a response; otherwise the command string is as shown in the headings in this section. For additional information on any Common Commands, refer to the *IEEE Standard 488.2-1987* (see "Related Documentation" in the front of this manual for more information on this standard).

Table 6-3. IEEE 488.2 Common Commands Functional Groupings

Category	Command	Title
General	*IDN	Identification Query
	*RST	Reset Command
	*TST?	Self-Test Query
Instrument Status	*CLS	Clear Status Command
	*ESE < mask >	Standard Event Status Enable Command
	*ESE?	Standard Event Status Enable Query
	*ESR?	Standard Event Status Register Query
	*PSC	Power-On Status Clear Command
	*PSC?	Power-On Status Clear Query
	*SRE < mask >	Service Request Enable Command
	*SRE?	Service Request Enable Query
	*STB?	Status Byte Query
	Macros	*DMC < name > , < cmds >
*EMC < state >		Enable Macros Command
*EMC?		Enable Macro Query
*GMC? < name >		Get Macro Query
*LMC?		Learn Macro Query
*PMC		Purge all Macros Command
*RMC < name >		Remove individual Macro Command
Synchronization	*OPC	Operation Complete Command
	*OPC?	Operation Complete Query
	*WAI	Wait-to-Continue Command

***ESE?** Standard Event Status Enable Query. Returns the weighted sum of all enabled (unmasked) bits in the Standard Event Status Register.

Example	10 OUTPUT 70900;"*ESE?"	<i>Sends status enable query</i>
	20 ENTER 70900;A	<i>Places response in variable</i>
	30 PRINT A	<i>Prints response</i>
	40 END	

***ESR?** Standard Event Status Register Query. Returns the weighted sum of all set bits in the Standard Event Status Register. After reading the register, *ESR? clears the register. The events recorded in the Standard Event Status Register are independent of whether or not those events are enabled with the *ESE command.

Example	10 OUTPUT 70900;"*ESR?"	<i>Sends Standard Event Status Register query</i>
	20 ENTER 70900;A	<i>Places response in variable</i>
	30 PRINT A	<i>Prints response</i>
	40 END	

***GMC? < name_string >** Get Macro Query. Returns *arbitrary block response data* which contains the command or command sequence defined by *name_string*. The command sequence will be prefixed with characters which indicate the number of characters that follow the prefix.

Example	10 OUTPUT 70900;"*GMC? 'LIST'"	<i>ask for definition of macro from *DMC example</i>
	20 ENTER 70900;Cmds\$	<i>enter into Cmds\$ the definition of the macro "LIST"</i>
	30 PRINT Cmds\$	<i>Cmds\$ = #214VXI:CONF:DLIS?</i>
	40 END	

In this case, the prefix consists of "#214". The 2 says to expect two character-counting digits. The 14 says that 14 characters of data follow. Had the returned macro been shorter, such as #15*EMC?, we would read this as 1 counting digit indicating 5 data characters.

***IDN?** Identity. Returns the device identity. The response consists of the following four fields (fields are separated by commas):

- Manufacturer
- Model Number
- Serial Number (returns 0 if not available)
- Firmware Revision (returns 0 if not available)

The *IDN? command returns the following command string for the E1405A:

```
HEWLETT-PACKARD,E1405A,0,A,01.00
```

This command will return the following string for the E1405B:

```
HEWLETT-PACKARD,E1405B,0,A,01.00
```

NOTE

The revision will vary with the revision of the ROM installed in the system. This is the only indication of which version of ROM is in the box. The major number (01 in the examples) indicates whether there have been functional changes made in this ROM. The minor number (00 in the examples) indicates whether only bug fixes and minor changes were made.

Example Get the ID fields from the system and print them.

10 DIM A\${50}	<i>Dimension array for ID fields</i>
20 OUTPUT 70900;"*IDN?"	<i>Queries identity</i>
30 ENTER 70900;A\$	<i>Places ID fields in array</i>
40 PRINT A\$	<i>Print ID fields</i>
50 END	

***LMC?** Learn Macros Query. Returns a quoted string *name* for each currently defined macro. If more than one macro is defined, the quoted strings are separated by commas (.). If no macro is defined, then a quoted null string ("") is returned.

***LRN?** Learn query command. *LRN? causes the instrument to respond with a string of SCPI commands which define the instrument's current state. Your application program can enter the *LRN? response data into a string variable, later to be sent back to the instrument to restore that configuration.

Example response from an HP E1326B voltmeter in the power-on state:

```
*RST;;CAL:ZERO:AUTO 1; :CAL:LFR +60; VAL +0.00000000E+000;
:DISP:MON:STAT 0; CHAN (@0); :FORM ASC, +7; :FUNC "VOLT";
:MEM:VME:ADDR +2097152; SIZE +0; STAT 0; :RES:APER
+1.666667E-002; OCOM 0; RANG +1.638400E+004; RANG:AUTO
1;;VOLT:APER +1.666667E-002; RANG +8.000000E+000; RANG:AUTO
1; :TRIG:COUN +1; DEL +0.00000000E+000; DEL:AUTO 1; :TRIG:SOUR
IMM; :SAMP:COUN +1; SOUR IMM;TIM +5.000000E-002 S
```

NOTE

The System Instrument no longer implements the *LRN? command. Attempting to have the System Instrument execute this command will generate an error -113 "Undefined header".

- *OPC** Operation Complete. Causes an instrument to set bit 0 (Operation Complete Message) in the Standard Event Status Register when all pending operations have been completed. By enabling this bit to be reflected in the Status Byte Register (*ESE 1 command), you can ensure synchronization between the instrument and an external computer or between multiple instruments. (Refer to "Synchronizing an External Computer and Instruments" earlier in this chapter for an example).
- *OPC?** Operation Complete Query. Causes an instrument to place an ASCII 1 into the instrument's output queue when all pending instrument operations are finished. By requiring the computer to read this response before continuing program execution, you can ensure synchronization between one or more instruments and the computer. (Refer to "Synchronizing an External Computer and Instruments" earlier in this chapter for an example).
- *PMC** Purge Macros Command. Purges all currently defined macros in the selected instrument.
- *PSC <flag>** Power-on Status Clear Command. Controls the automatic power-on clearing of the Service Request Enable register and Standard Event Status Enable register. Executing *PSC 1 disables any previously enabled bits at power-on, preventing the System Instrument from requesting service when power is cycled. Executing *PSC 0 causes any previously enabled bits to remain enabled at power-on which allows the System Instrument to request service (if it has been enabled - *SRE) when power is cycled. The value of *flag* is stored in non-volatile memory.

Example This example configures the System Instrument to request service from the external computer whenever power is cycled.

Status Byte register and Standard Event Status register bits remain enabled (unmasked) after cycling power

10 OUTPUT 70900;"*PSC 0"

Enable bit 5 (Standard Event Status Register Summary Bit) in the Status Byte Register

20 OUTPUT 70900;"*SRE 32"

Enable bit 7 (Power-on bit) in the Standard Event Status Register to be reflected as bit 5 in the Status Byte Register

30 OUTPUT 70900;"*ESE 128"

- *PSC?** Power-on status clear query. Returns a response indicating whether an instrument's Status Byte Register and Standard Event Status Register bits remain enabled or become disabled at power-on. A "1" means the bits are disabled at power-on; a "0" means the bits remain enabled at power-on.

***RMC <name_string >** Remove Individual Macro Command. Purges an individual macro identified by the *name_string* parameter.

Example output 70900;"*RMC 'LIST'" *remove macro command from *DMC example*

NOTE: At printing time, *RMC is a command proposed for a revision and re-designation of ANSI/IEEE Std 488.2-1987.

***RST** Reset. Resets an instrument as follows:

- Sets the instrument to a known state (usually the power-on state)
- Aborts all pending operations
- Disables the *OPC and *OPC? modes.

*RST does not affect:

- The state of the HP-IB interface
- The HP-IB address
- The output queue
- The Service Request Enable Register
- The Standard Event Status Enable Register
- The power-on flag
- Calibration data
- Protected user data

***SRE <mask >** Service Request Enable. When a service request event occurs, it sets a corresponding bit in the Status Byte Register (this happens whether or not the event has been enabled (unmasked) by *SRE). The *SRE command allows you to identify which of these events will assert an HP-IB service request (SRQ). When an event is enabled by *SRE and that event occurs, it sets a bit in the Status Byte Register and issues an SRQ to the computer (sets the HP-IB SRQ line true). You enable an event by specifying its decimal weight for <mask>. To enable more than one event, specify the sum of the decimal weights. Refer to "The Status Byte Register" earlier in this chapter for a table showing the contents of the Status Byte Register.

Example OUTPUT 70900;"*SRE 160" *Enables bits 5 & 7. Respective weights are 32 + 128 = 160*

***SRE?** Status Register Enable Query. Returns the weighted sum of all enabled (unmasked) events (those enabled to assert SRQ) in the Status Byte Register.

Example 10 OUTPUT 70900;"*SRE?" *Sends Status Register Enable query*
 20 ENTER 70900;A *Places response in variable*
 30 PRINT A *Prints response*
 40 END

***STB?** Status Byte Register Query. Returns the weighted sum of all set bits in the Status Byte Register. Refer to "The Status Byte Register" earlier in this chapter for a table showing the contents of the Status Byte Register.

Comments You can read the Status Byte Register using either the *STB? command or an HP-IB serial poll (IEEE 488.1 message). Both methods return the weighted sum of all set bits in the register. The difference between the two methods is that *STB? does not clear bit 6 (Service Request); serial poll does clear bit 6. No other status byte register bits are cleared by either method with the exception of the Message Available bit (bit 4) which may be cleared as a result of reading the response to *STB?.

Example	10 OUTPUT 70900;"*STB?"	<i>Sends Status Byte Register query</i>
	20 ENTER 70900;A	<i>Places response in variable</i>
	30 PRINT A	<i>Prints response</i>
	40 END	

***TST?** Self-Test. Causes an instrument to execute an internal self-test and returns a response showing the results of the self-test. A zero response indicates that self-test passed. A value other than zero indicates a self-test failure or error.

Example	10 OUTPUT 70900;"*TST?"	<i>Execute self-test, return response</i>
	20 ENTER 70900;A	<i>Places self-test response in variable</i>
	30 PRINT A	<i>Prints response</i>
	40 END	

***WAI** Wait-to-continue. Prevents an instrument from executing another command until the operation caused by the previous command is finished (sequential operation). Since all instruments normally perform sequential operations, executing the *WAI command causes no change to the instrument's operation.

HP-IB Message Reference

This section describes IEEE-488.1 defined messages and their affect on instruments installed in the mainframe. The examples shown are specifically for HP 9000 Series 200/300 computers using BASIC language. Although any IEEE-488 controller can send these messages, the syntax may be different from that shown here.

Go To Local (GTL)

Places an instrument in local state.

Comments

- Refer to the Local Lockout message, later in this chapter, for information on how GTL affects front panel lockout.

Examples

LOCAL 7

Sets HP-IB remote enable line false (all instruments go to local). (You must now execute REMOTE 7 to return to remote mode).

LOCAL 70900

Issues HP-IB GTL to System Instrument. (The instrument will return to remote mode when it is listen addressed.)

Group Execute Trigger (GET)

Executing a group execute trigger will trigger an instrument assuming the following conditions are true:

- The instrument's trigger source is set to Bus (TRIG:SOUR BUS command), and:
- The instrument is in the Wait For Trigger state, and:
- The instrument is addressed to listen (can be done by sending any command, the REMOTE 709ss (ss = secondary address) command, or with the LISTEN command).

Comments

- For instruments in the servant area of an E1405 Command Module, only one instrument at a time can be programmed to respond to GET. This is because only one instrument can be addressed to listen at any one time. GET has no affect on the System Instrument.

Interface Clear (IFC)

Unaddresses all instruments in the servant area of the specified Command Module and breaks any bus handshaking in progress.

Example

ABORT 7

Device Clear (DCL) or Selected Device Clear (SDC)

DCL clears all instruments in the Command Module servant area. SDC clears a specific instrument. The purpose of DCL or SDC is to prepare one or more instruments to receive and execute commands (usually *RST). DCL or SDC do the following to each instrument:

- Clear the input buffer and output queue.
- Reset the command parser.
- Disable any operation that would prevent *RST from being executed.
- Disable the Operation Complete and Operation Complete Query modes.

DCL or SDC do not affect:

- Any settings or stored data in the instrument (except the Operation Complete and Operation Complete Query modes)
- Front panel operation
- Any instrument operation in progress (except as stated above)
- The status byte (except for clearing the Message Available bit as a result of clearing the output queue).

Examples

CLEAR 7

Clears all instruments

CLEAR 70900

Clears the System Instrument

Local Lockout (LLO)

When an instrument is in remote mode, Local Lockout prevents an instrument from being operated from the mainframe's front panel.

Comments

- Certain front panel operations such as menu control and display scrolling are still active in Local Lockout mode.
- If the instrument is in the local state when you send LOCAL LOCKOUT, it remains in local. If the instrument is in the remote state when you send LOCAL LOCKOUT, front panel control is disabled immediately for that instrument.
- After executing LOCAL LOCKOUT, you can enable the keyboard by sending the LOCAL 7 command or by cycling power. The LOCAL 709ss (ss = secondary address) command enables the front panel for that instrument but a subsequent remote command disables it. Sending the LOCAL 7 command removes lockout for all instruments and places them in the local state.

Examples

10 REMOTE 70900

Sets the System Instrument remote state

20 LOCAL LOCKOUT 7

Disables front panel control for the System Instrument and all other instruments that were in the remote state.

30 END

Remote Sets the HP-IB remote enable line (REN) true which places an instrument in the remote state.

- Comments**
- The REMOTE 709ss (ss = secondary address) command places the instrument in the remote state. The REMOTE 7 command, does not, by itself, place the instrument in the remote state. After sending the REMOTE 7 command, the instrument will only go into the remote state when it receives its listen address.
 - In most cases, you will only need the REMOTE command after using the LOCAL command. REMOTE is independent of any other HP-IB activity and toggles a single bus line called REN. Most controllers set the REN line true when power is applied or when reset.

Examples

REMOTE 7	<i>Sets HP-IB REN line true</i>
REMOTE 70900	<i>Sets REN line true and addresses System Instrument</i>

Serial Poll (SPOLL) The SPOLL command, like the *STB? Common Command, returns the weighted sum of all set bits in an instrument's Status Byte Register (status byte). Refer to "The Status Byte Register" earlier in this chapter for a table showing the contents of the Status Byte Register.

- Comments**
- The SPOLL command differs from the *STB? command in that SPOLL clears bit 6 (RQS). Executing *STB? does not clear bit 6.

Examples

10 P = SPOLL (70900)	<i>Sends Serial Poll, places response into P</i>
20 DISP P	<i>Displays response</i>
30 END	

Command Quick Reference

The following tables summarize SCPI and IEEE 488.2 Common (*) commands for the HP E1405 Command Module System Instrument.

SCPI Commands Quick Reference	
Command	Description
DIAGnostic	
:BOOT	
:COLD	Restarts System processor, clears stored configurations.
[:WARM]	Same as cycling power.
:COMMunicate	
:SERial[0]	
[:OWNer] [SYSTem IBASic NONE]	Allocates the built-in serial interface.
[:OWNer]?	Returns SYST, IBAS, or NONE.
:SERial[n]	
:STORe	Stores serial communication parameters into non-volatile storage.
:DOWNload	
:CHECked	
[:MADDress]	Write data to non-volatile user RAM starting at the specified address using error correction.
:SADDress	Write data to non-volatile user RAM at the specified address using error correction.
[:MADDress] <address>, <data>	Write data to non-volatile user RAM starting at the specified address.
:SADDress <address>, <data>	Write data to non-volatile user RAM at the specified address.
:DRAM	
:AVailable?	Returns the amount of RAM remaining in the DRAM (Driver RAM) segment.
:CREate <size> <num_drivers>	Creates a non-volatile RAM area for loading instrument drivers.
:DRIVer	
:LOAD <driver_block>	Loads the instrument driver contained in the specified driver_block into a previously created DRAM segment.
:LOAD	
:CHECked	Loads the instrument driver contained in the specified driver_block into a previously created DRAM segment using error correction.
:LIST	
[:ALL]	Lists all drivers from all driver tables (RAM and ROM) found on the system.
:RAM	Lists all drivers found in the RAM driver table.
:ROM	Lists all drivers found in the ROM driver table.
:INTerrupt	
:ACTivate [ON OFF 1 0]	Enable VXIbus interrupt acknowledgement.
:SETup[n] [ON OFF 0 1]	Enables or disables System Instrument control of VXI interrupt line [n].
:SETup[n]?	Returns current state of SETUP[n].
:PRIority[n] [<priority> MIN MAX DEF]	Specifies the priority level of VXI interrupt line [n].
:PRIority[n]? [MIN MAX DEF]	Returns priority level of VXI interrupt line [n].
:RESPonse?	Returns response from the highest priority interrupt line.
:NRAM	
:ADDRess?	Returns starting address of the User non-volatile RAM.
:CREate <size> MIN MAX	Creates a User non-volatile RAM segment.

SCPI Commands Quick Reference	
Command	Description
<pre> :CREate? [MIN MAX] :PEEK? < address> MIN MAX, < width> :POKE < address> MIN MAX, < width>, < data> :RDISk :ADDRes? :CREate < size> MIN MAX :CREate? [MIN MAX] :UPLoad [:MADDress]? < address>, < byte_count> :SADDress? < address>, < byte_count> </pre>	<p>Returns the current or allowable size of User NVRAM.</p> <p>Returns an 8, 16, or 32 bit value from memory.</p> <p>Stores an 8, 16, or 32 bit value to RAM.</p> <p>Returns the starting address of an IBASIC RAM volume.</p> <p>Allocates RAM for an IBASIC RAM volume.</p> <p>Returns the current or allowable size of the RAM vol.</p> <p>Returns data from non-volatile user RAM starting at address.</p> <p>Returns data from non-volatile user RAM at address.</p>
<p>OUTPut</p> <pre> ECLTrg < n> (:ECLTrg0 or :ECLTrg1) :IMMediate :LEVel [:IMMediate] 1 0 ON OFF :LEVel? :SOURce INT EXT NONE :SOURce? [:STATe] 1 0 ON OFF :STATe? :EXTernal :IMMediate :LEVel [:IMMediate] 1 0 ON OFF :LEVel? :SOURce INT TTL ECLT NONE :SOURce? :STATe 1 0 ON OFF :STATe? :TTLTrg < n> (:TTLTrg0 through :TTLTrg7) :IMMediate :LEVel [:IMMediate] 1 0 ON OFF :LEVel? :SOURce INT EXT NONE :SOURce? [:STATe] 1 0 ON OFF :STATe? </pre>	<p>Generate pulse on specified ECL trigger line.</p> <p>Sets the output level of the specified ECL trigger line.</p> <p>Returns the output level of the specified ECL trigger line.</p> <p>Set the source which drives the selected ECL trigger line.</p> <p>Returns the source driving the selected ECL trigger line.</p> <p>Enables configuration of the specified ECL trigger line.</p> <p>Returns the current state of the selected trigger line.</p> <p>Generate pulse on Command Module "Trig Out" port.</p> <p>Sets the output level of the "Trig Out" port.</p> <p>Returns the output level of the "Trig Out" port.</p> <p>Sets the source which drives the "Trig Out" port.</p> <p>Returns the source driving the "Trig Out" port.</p> <p>Enables configuration of the "Trig Out" port.</p> <p>Returns the state of the "Trig Out" port.</p> <p>Generate pulse on the selected TTLTrg trigger line.</p> <p>Sets the output level of the selected TTLTrg trigger line.</p> <p>Returns the output level of the selected TTLTrg trigger line.</p> <p>Sets the source which drives the selected TTLTrg trigger line.</p> <p>Returns the source driving the selected TTLTrg trigger line.</p> <p>Enables configuration of the selected TTLTrg trigger line.</p> <p>Returns the state of the selected TTLTrg trigger line.</p>
<p>STATus</p> <pre> :OPERation :CONDition? :ENABle 256 :ENABle? [:EVENTi]? :PRESet </pre>	<p>Returns the state of the condition register.</p> <p>Set Standard Operation Enable Register mask.</p> <p>Returns value of enable mask.</p> <p>Returns value of the bit set in the Event register (Standard Operation Status Group).</p> <p>Presets status registers</p>

SCPI Commands Quick Reference

Command	Description
:QUESTIONable	Always returns +0.
:CONDition?	Set Questionable Status Register enable mask.
:ENABle <mask>	Returns value of enable mask.
:ENABle?	Always returns +0.
[:EVENT]?	
SYSTEM	
:COMMunicate	
:GPIB	Returns GPIB address or min max allowed value.
:ADDRes?	
:SERial[n]	
:CONTRol	
:DTR ON OFF STANdard IBFull	Sets mode for modem control line DTR.
:DTR?	Returns current mode of DTR line.
:RTS ON OFF STANdard IBFull	Sets mode for modem control line RTS.
:RTS?	Returns current mode of RTS line.
[:RECeive]	
:BAUD <baud_rate> MIN MAX	Sets transmit and receive baud rate of serial interface.
:BAUD? [MIN MAX]	Returns the current or allowable baud setting.
:BITS 7 8 MIN MAX	Sets the number of data bits in the serial data frame.
:BITS? [MIN MAX]	Returns the current or allowable BITS setting.
:PACE	
[:PROTocol] XON NONE	Sets the receive pacing protocol to XON/XOFF or none.
[:PROTocol]?	Returns the state of receive pacing protocol.
:THReshold	
:STARt <char_count>	Sets the input buffer start threshold for input pacing.
:STARt? [MIN MAX]	Returns current or allowable STARt threshold level.
:STOP <char_count>	Sets the input buffer stop threshold for input pacing.
:STOP? [MIN MAX]	Returns the current or allowable STOP threshold level.
:PARity	
:CHECK 1 0 ON OFF	Enables/disables receive parity checking.
:CHECK?	Returns the current state of receive parity checking.
[:TYPE] EVEN ODD ZERO ONE NONE	Sets the type of receive and transmit parity.
[:TYPE]?	Returns the current parity type setting.
:SBITs 1 2 MIN MAX	Sets the number of stop bits for receive and transmit.
:SBITs? MIN MAX	Returns the number of stop bits set.
:TRANsmit	Note: HP E1324A is always ...TRAN:AUTO ON
:AUTO 1 0 ON OFF	Links/unlinks the transmit and receive pacing protocol.
:AUTO?	Returns the current transmit/receive pacing linkage.
:PACE	
[:PROTocol] XON NONE	Sets the transmit pacing protocol to XON/XOFF or none.
[:PROTocol]?	Returns the state of transmit pacing protocol.
:DATE <year>,<month>,<day>	Sets system calendar.
:DATE? [MIN MAX,MIN MAX,MIN MAX]	Returns current date or min max allowable values.
:ERRor?	Returns oldest error message in Error Queue.
:TIME <hour>,<minute>,<second>	Sets the system clock.
:TIME? [MIN MAX,MIN MAX,MIN MAX]	Returns current time or min max allowable values.
:VERSion?	Returns SCPI version for which this instrument complies.

SCPI Commands Quick Reference	
Command	Description
VXI	
:CONFigure	
:CTABLE <address>	Links the Commander/Servant Hierarchy table to the Command Module (resource manager) processor.
:CTABLE?	Gets the Commander/Servant Hierarchy table starting address
:DCTable <address>	Links the Dynamic Configuration table to the Command Module (resource manager) processor.
:DCTable?	Gets the Dynamic Configuration table starting address.
:DeviceLADD?	Returns a list of the logical addresses in the system.
:DeviceLIS?	Returns information about one or all installed devices.
:DeviceNUMBER?	Returns the number of installed devices.
:ETABLE	Links the Extender Device table to the Command Module (resource manager) processor.
:ETABLE?	Gets the Extender Device table starting address.
:HIERarchy	Gets the current hierarchy configuration data for the selected logical address (see VXI:SELEct)
:ALL?	Gets the current hierarchy configuration data for all logical addresses.
:INformation	Gets the static information about the selected logical address (see VXI:SELEct).
:ALL?	Gets the static information about all logical addresses.
:ITABLE <address>	Links the Interrupt Line Allocation table to the Command Module (resource manager) processor.
:ITABLE?	Gets the Interrupt Line Allocation table starting address.
:LADDRESS	
:MEXTender?	Gets a comma separated list of all logical addresses for mainframe extenders in the system when issued to a Resource Manager. Generates an error if received by a device other than the Resource manager.
:LADDRESS?	Gets a comma separated list of all logical addresses of devices in the system when issued to a Resource Manager. Generates an error if received by a device other than the Resource manager.
:MEXTender	
:ECLTrg[n] <IN OUT NONE>	Configures the selected mainframe extender to direct the ECL trigger specified by [n].
:INTerrupt[n] <IN OUT NONE>	Configures the selected mainframe extender to direct the interrupt line specified by [n].
:TTLTrg[n] <IN OUT NONE>	Configures the selected mainframe extender to direct the TTL trigger specified by [n].
:MTABLE <address>	Link A24/A32 Address Allocation Table to Command Module (resource manager) processor.
:MTABLE?	Query A24/A32 Address Allocation table starting address.
:NUMBER	
:MEXTender	Gets the number of mainframe extenders in the system when issued to a Resource Manager. Generates an error if received by a device other than the Resource manager.
:NUMBER?	Gets the number of devices in the system when issued to a Resource Manager. Generates an error if received by a device other than the Resource manager.
:QUERy? <logical_addr>	Read Data Low register of device at logical_addr.
:READ? <logical_addr>, <register_num>	Read the contents of the device register at register_num.
:RECeive	
[:MESSAge]? <logical_addr> [, <end_of_msg>]	Receive message from message-based device at logical_addr.
:REGister	

SCPI Commands Quick Reference

Command	Description
:READ? <numeric_value <reg_name>	Returns the contents of the specified 16 bit register at the selected logical address (see VXI:SElect).
:WRITe <numeric_value <reg_name>,<data>	Writes to the specified 16 bit register at the selected logical address (see VXI:SElect).
:RESet <logical_addr>	Resets the device at the specified logical addr.
:RESet?	Resets the device at the selected logical address (see VXI:SElect).
:ROUte	
:ECLTrg[n]	Sets the direction for the specified trigger line in all mainframe extenders so that the device selected by VXI:SEL can source the trigger and all other devices in the system can monitor it.
:INTerrupt[n]	Sets the direction for the specified interrupt line in all mainframe extenders so that the device selected by VXI:SEL can handle the interrupts and all other devices in the system can monitor it.
:TTLTrg[n]	Sets the direction for the specified trigger line in all mainframe extenders so that the device selected by VXI:SEL can source the trigger and all other devices in the system can monitor it.
:SElect <numeric_value>	Specifies the logical address to be used by all subsequent commands in the VXI subsystem.
:SEND	
:COMMand <logical_addr>,<command>[,<data>]	Send word serial command to device at logical_addr.
:COMMand? <logical_addr>,<command> [,<data1>[,<data2>]]	Send word serial command to device at logical_addr and then wait for response from Data Low register.
:MESSAge <logical_addr>,<"msg_string">[,<end_flag>]	Send command to message-based device at logical_addr.
:WRITe <logical_addr>,<register_num>,<data>	Write data to the device register at logical_addr.
:WSProtocol	
:COMMand [:ANY] <cmd_word>	Sends cmd_word as a word serial command to the logical address set using VXI:SEL.
:AHLIne <hand_id>,<line_#>	Assigns a handler to the logical address set using VXI:SEL. A line number of 0 means the handler is to be disconnected.
AIlIne <int_id>,<line_number>	Assigns an interrupter line to the logical address set using VXI:SEL. A line number of 0 means the handler is to be disconnected.
:AMControl <response_mask>	Sends an Asynchronous Mode Control command to the logical address set using VXI:SEL.
:ANO	Sends an Abort Normal Operation command to the logical address set using VXI:SEL.
:BAVailable	Sends a Byte Available command to the logical address set using VXI:SEL.
:BNO	Sends a Begin normal Operation command to the logical address set using VXI:SEL.
:BRQ	Sends a Byte Request command to the logical address set using VXI:SEL.
:CEVent	Sends a Control event command to the logical address set using VXI:SET.
:CLR	Sends a Clear command to the logical address set using VXI:SEL.
:CLOCK	Sends a Clear lock command to the logical address set using VXI:SEL.
:CRESPonse	Sends a control Response command to the logical address set using VXI:SEL.
:ENO	Sends an End Normal Operation command to the logical address set using VXI:SEL.

SCPI Commands Quick Reference	
Command	Description
:GDEvice	Sends a Grant Device command to the logical address set using VXI:SEL.
:ICOMmander	Sends an Identify Commander command to the logical address set using VXI:SEL.
:RDEvice	Sends a Release Device command to the logical address set using VXI:SEL.
:RHANdlers	Sends a Read Handlers command to the logical address set using VXI:SEL.
:RHLine	Sends a Read Handler Line command to the logical address set using VXI:SEL.
:RILine	Sends a Read Interrupter Line command to the logical address set using VXI:SEL.
:RINTerrupter	Sends a Read Interrupter command to the logical address set using VXI:SEL.
:RMODid	Sends a Read MODID command to the logical address set using VXI:SEL.
:RPERror	Sends a Read Protocol Error command to the logical address set using VXI:SEL.
:RPRotocol	Sends a Read Protocol command to the logical address set using VXI:SEL.
:RSTB	Sends a Read Status Byte command to the logical address set using VXI:SEL.
:RSARea	Sends a Read Servant Area command to the logical address set using VXI:SEL.
:SLModid	Sends a Set Lower MODID command to the logical address set using VXI:SEL.
:SLOCK	Sends the Set Lock command to the logical address set using VXI:SEL.
:SUModid	Sends a Set Upper MODID command to the logical address set using VXI:SEL.
:TRIGger	Sends a Trigger command to the logical address set using VXI:SEL.
:MESSage	
:RECCeive?	Receives a message from the logical address set using VXI:SEL using both the word serial protocol and the byte transfer protocol.
:SEND	Sends a message to the logical address set using VXI:SEL. The message is sent using both the word serial protocol and the byte transfer protocol.
:QUERy	
[:ANY?]	Sends cmd_word as a word serial command to the logical address set using VXI:SEL and waits for return value.
:AHLIne? <hand_id>, <line_#>	Assigns a handler to the logical address set using VXI:SEL and waits for a response. A line number of 0 means the handler is to be disconnected.
:AILIne? <int_id>, <line_#>	Assigns an interrupter line to the logical address set using VXI:SEL and waits for a response. A line number of 0 means the handler is to be disconnected.
:AMControl?	Sends an Asynchronous Mode Control command to the logical address set using VXI:SEL and waits for a response.
:ANO?	Sends an Abort Normal Operation command to the logical address set using VXI:SEL and waits for a response.
:BNO?	Sends a Begin normal Operation command to the logical address set using VXI:SEL and waits for a response.
:BRQ?	Sends a Byte Request command to the logical address set using VXI:SEL and waits for a response.
:CEVEnt?	Sends a Control event command to the logical address set using VXI:SEL and waits for a response.

SCPI Commands Quick Reference	
Command	Description
:CRESPonse?	Sends a control Response command to the logical address set using VXI:SEL and waits for a response.
:ENO?	Sends an End Normal Operation command to the logical address set using VXI:SEL and waits for a response.
:RDEvice?	Sends a Release Device command to the logical address set using VXI:SEL and waits for a response.
:RHANdlers?	Sends a Read Handlers command to the logical address set using VXI:SEL and waits for a response.
:RHLine?	Sends a Read Handler Line command to the logical address set using VXI:SEL and waits for a response.
:RLLine?	Sends a Read Interrupter Line command to the logical address set using VXI:SEL and waits for a response.
:RINTerrupter?	Sends a Read Interrupter command to the logical address set using VXI:SEL and waits for a response.
:RMODid?	Sends a Read MODID command to the logical address set using VXI:SEL and waits for a response.
:RPERror?	Sends a Read Protocol Error command to the logical address set using VXI:SEL and waits for a response.
:RPRotocol?	Sends a Read Protocol command to the logical address set using VXI:SEL and waits for a response.
:RSTB?	Sends a Read Status Byte command to the logical address set using VXI:SEL and waits for a response.
:RSArea?	Sends a Read Servant Area command to the logical address set using VXI:SEL and waits for a response.
:SLModid?	Sends a Set Lower MODID command to the logical address set using VXI:SEL and waits for a response.
:RESPonse?	Retrieves the response (one word of integer data) resulting from a WSPRotocol:COMMand command.

IEEE 488.2 Common Commands Quick Reference		
Category	Command	Title
General	*IDN?	Identification Query
	*RST	Reset Command
	*TST?	Self Test Query
Instrument Status	*CLS	Clear Status Command
	*ESE <mask>	Standard Event Status Enable Register Command
	*ESE?	Standard Event Status Enable Query
	*ESR?	Standard Event Status Register Query
	*PSC <flag>	Power-on Status Clear Command
	*PSC?	Power-on Status Clear Query
	*SRE <mask>	Service Request Enable Command
	*SRE?	Service Request Enable Query
	*STB?	Status Byte Register Query
Macros	*DMC <name>,<cmd_data>	Define Macro Command
	*EMC <enable>	Enable Macro Command
	*EMC?	Enable Macro Query
	*GMC? <name>	Get Macro Query
	*LMC?	Learn Macro Query
	*PMC	Purge all Macros Command
	*RMC <name>	Remove individual Macro Command
Synchronization	*OPC	Operation Complete Command
	*OPC?	Operation Complete Query
	*WAI	Wait-to-Continue Command

Specifications

Real Time Clock Accuracy: 0.005% of elapsed time since last set.

Temperature coefficient: +0.001%, -0.012% of time since last set (per °C change in temperature).

Resolution: 1.0 sec

Non-volatile lifetime: 10 months typical for a module with 512Kb memory (following a 15 hour battery charge). 5 months typical for a module with 1Mb of memory.

CLK10 Input: TTL or low level AC

Minimum input level: ± 40 mVp-p

Maximum input level: ± 42.5 Vp-p

Output: TTL

Jitter: 0.03% (-55dB)

Initial Accuracy: 50 ppm

Trigger Input TTL levels

Input load: 5 k Ω , 50 pF

Maximum Rate: 12.5 MHz (TTL), 40 MHz (ECL)

Minimum pulse width: 30 nsec (TTL), 12.5 nsec (ECL)

Maximum trigger delay: 30 nsec

Memory 128 kBytes user accessible, non-volatile RAM on a module with 512Kb of memory. Memory is expandable to 1Mb (640Kb user accessible). NiCad battery backed (10 month typical lifetime for modules with 128Kb of user accessible RAM and 5 months typical for modules with 640Kb of user accessible RAM, following a 15 hour battery charge).

Power Requirement

DC Volts	DC Current	Dynamic Current
+5V	2.1A	0.1A
+12V	0.01A	0.005A
-12V	0.01A	0.005A
-5.2V	0.4A	0.04A
-2V	0.05A	0.005A

Cooling Requirements For 10 °C rise 1.5 liters/second 0.4mm H_2O

SCPI Conformance Information

The HP E1300/1301A conforms to SCPI-1990.0

In documentation produced prior to June 1990, these SCPI commands are labeled as TMSL commands.

The following tables list all the SCPI conforming, approved, and non-SCPI commands that the HP E1405B can execute. Individual commands may not execute without having the proper plug-in module installed in the HP E1300/13301A. Each plug-in module manual describes the commands that apply to that module.

Switchbox Configuration

The following plug-in modules can be configured as switchbox modules. Refer to the individual plug-in User's Manual for configuration information.

HP E1345A	HP E1353A	HP E1366A
HP E1346A	HP E1357A	HP E1367A
HP E1347A	HP E1358A	HP E1368A
HP E1351A	HP E1361A	HP E1369A
HP E1352A	HP E1364A	HP E1370A

Table A-1. Switchbox SCPI-1990.0 Confirmed Commands

ABORt	STATUs
	:QUEStionable
ARM	:CONDItion?
:COUNt	[:EVENt]?
	:ENABle
INITiate	:ENABle?
[:IMMediate]	:OPERation
:CONTInous	:CONDItion?
	[:EVENt]?
OUTPut	:ENABle
:ECLTrg	:ENABle?
[:STATe]	:PRESet
:TTLTrg	
[:STATe]	SYSTem
	:ERRor?
[ROUTe]	:CPON
:OPEN	:CTYPe?
:OPEN?	:VERSion?
:CLOSe	
:CLOSe?	TRIGger
:SCAN	[:IMMediate]
	:SOURce
	:SLOPe

Table A-2. Switchbox Non-SCPI Commands

DISPlay	[ROUTe]
:MONitor	:SCAN
[:STATe]	[:LIST]
:CARD	:MODE
	:PORT
SYSTem	:SETTing
:CDEscription?	[:TIME]
	:TIME?

Multimeter Commands The following tables apply to the HP E1326A and E1326B.

Table A-3. Multimeter SCPI-1990.0 Confirmed Commands

ABORt	[SENSe]
CALibration	:FUNction
:ZERo	:FUNction?
:AUTo	:RESistance
:AUTo?	:APERture
:VALue	:APERture?
	:RANGe
	:AUTo
CONFigure	:AUTo?
:FREStance	:RANGe?
:RESistance	:RESolution
:TEMPerature	:RESolution?
:VOLTag	:VOLTag
:AC	:AC
[:DC]	:RANGe
	:RANGe?
CONFigure?	[:DC]
	:RANGe
FETCh?	:AUTo
	:AUTo?
FORMat	:RANGe?
[:DATA]	:RESolution
	:RESolution?
INITiate	
[:IMMediate]	
MEASure	STATus
:FREStance?	:QUESTionable
:RESistance?	:CONDition?
:TEMPerature?	[:EVENTi]?
:VOLTag	:ENABle
:AC?	:ENABle?
[:DC]?	:OPERation
	CONDition?
	[:EVENTi]?
	:ENABle
	:ENABle?
READ?	:PREset
	SYSTem
	:ERRor?
	:CTYPe?
	:VERsion?
	TRIGger
	:COUNt
	:COUNt?
	:DELay?
	:AUTo
	:AUTo?
	:DELay?
	[:IMMediate]
	:SOURce
	:SOURce?

Table A-4. Multimeter SCPI Approved (not confirmed) Commands

[SENSe]
:RESistance
:NPLC
:NPLC?
:VOLtag
:NPLC
:NPLC?

Table A-5. Multimeter Non-SCPI Commands

CALibration	MEMory
:LFRequency	:VME
:LFRequency?	:ADDRess
:STRain	:ADDRess?
	:SIZE
CONFigure	:SIZE?
:STRain	:STATe
	:STATe?
:QUARter	
:HBENding	[ROUTe]
:HPOisson	:FUNction
:FBENding	
:FPOisson	SAMPle
:FBPOisson	:COUNT
:QTENsion	:COUNT?
:QCOMpression	:SOURce
:UNSTrained	:SOURce?
	:TIMer
DISPlay	:TIMer?
:MONitor	
:CHANnel	[SENSe]
:CHANnel?	:RESsistance
[:STATe]	:OCOMpensated
[:STATe]?	:OCOMpensated?
	:STRain
MEASure	:GFACtor
:STRain	:POISSon
:QUARter?	:UNSTrained
:HBENding?	
:HPOisson?	SYSTem
:FBENding?	:CDEscription
:FPOisson?	
:FBPOisson?	
:QTENsion?	
:QCOMpression?	
:UNSTrained?	

Counter Commands

The following tables apply to the HP E1332A 4 Channel Counter/Totalizer and the HP E1333A 3 Channel Universal Counter.

Table A-6. HP E1332A SCPI-1990.0 Confirmed Commands

ABORt	READ?
CONFIgure	[SENSe]
:FREQuency	:FUNcTion
:PERiod	:FREQuency
:PWIDth	:PERiod
:NWIDth	:FREQuency
CONFIgure?	:APERture
	:APERture?
FETCh?	STATus
FORMat	:QUEStionable
[:DATA]	[:EVENt]?
	:CONDition?
INITiate	:ENABle
[:IMMediate]	:ENABle?
	:OPERation
INPUt	[:EVENt]?
:FILTer	:CONDition?
[:LPASs]	:ENABle
[:STATe]	:ENABle?
[:STATe]?	:PREset
:FREQuency	SYSTEM
:FREQuency?	:ERRor?
	:VERsion?
MEASure	TRIGger
:FREQuency?	[:IMMediate]
:PERiod?	:SOURCe
:PWIDth?	:SOURCe?
:NWIDth	

Table A-7. HP E1332A Non-SCPI Commands

CONF[<channel >]	[SENSe[<channel >]]
:TOTalize	:PERiod
:TINTerval	:NPERiods
:UDCount	:NPERiods?
	:TOTalize
DISPlay	:GATE
:MONitor	[:STATe]
:CHANnel	[:STATe]?
:CHANnel?	:POLarity
[:STATe]	:POLarity?
[:STATe]?	:EVENt
	:LEVel
INPUt	:LEVel?
:ISOLate	:SLOPe
:ISOLate?	:SLOPe?
MEASure[<channel >]	
:TINTerval? >	

Table A-8. HP E1333A SCPI-1990.0 Confirmed Commands

ABORt	READ?
FETCh?	[SENSe]
CONFigure	:FUNctIon
:FREQuency	:FREQuency
:PERiod	:PERiod
:PWIDth	:FREQuency
:NWIDth	:APERture
	:APERture?
CONFigure?	STATUs
FORMat	:QUEStionable
[:DATA]	[:EVENT]?
	:CONDition?
	:ENABle
INITiate	:ENABle?
[:IMMediate]	:OPERation
	[:EVENT]?
INPut	:CONDition?
:ATTenuation	:ENABle
:ATTenuation?	:ENABle?
:COUPling	:PREset
:COUPling?	
:FILTer	
[:LPASs]	SYStem
	:ERRor?
[:STATe]	:VERsion?
[:STATe]?	
:IMPedance	TRIGger
:IMPedance?	[:IMMediate]
	:SOURCe
MEASure	:SOURCe?
:FREQuency?	
:PERiod?	
:PWIDth?	
:NWIDth?	

Table A-9. HP E1333A Non-SCPI Commands

CONF[< channel >]	[SENSe[< channel >]]
:TOTAlize	:PERiod
:TINTerval	:NPERiods
:RATio	:NPERiods?
	:RATio
DISPlay	:NPERiods
:MONitor	:NPERiods?
:CHANnel	:TINTerval
:CHANnel?	:NPERiods
[:STATe]	:NPERiods?
[:STATe]?	:EVENT
	:LEVel
MEASure[< channel >]	:LEVel?
:TINTerval?	:SLOPe
:RATio?	:SLOPe?

D/A Converter Commands

The following tables apply to the HP E1328A 4 Channel D/A Converter.

Table A-10. HP E1328A SCPI-1990.0 Confirmed Commands

CALibration	STATus
:STATe	:QUEStionable
:STATe?	:CONDition?
	[:EVENT]?
SYSTem	:ENABle
:ERRor?	:ENABle?
:VERSion?	:OPERation
	:CONDition?
	[:EVENT]?
	:ENABle
	:ENABle?

Table A-11. HP E1328A Non-SCPI Commands

CALibration	SOURce
:VOLTage	:VOLTage < channel >
:CURRent	:VOLTage < channel > ?
	:CURRent < channel >
DISPlay	:CURRent < channel > ?
:MONitor	:FUNCtion < channel > ?
:CHANnel	
:CHANnel?	
[:STATe]	
:STRing?	

Digital I/O Commands

The following tables apply to the HP E1330A Quad 8-bit Digital I/O Module.

Table A-12. HP E1330A SCPI-1990.0 Confirmed Commands

STATUS	SYSTEM
:QUESTIONable	:ERROR?
:CONDition?	:VERSion?
[:EVENT]?	
:ENABle	
:ENABle?	
:OPERation	
:CONDition?	
[:EVENT]?	
:ENABle	
:ENABle?	
:PREset	

Table A-13. HP E1330A Non-SCPI Commands

DISPlay	[SOURce]
:MONitor	:DIGital
[:STATe]	:TRACe
:PORT	:CATalog
:PORT?	[:DATA]
:STRing?	[:DATA]?
	:DEFine
	:DELete
MEASure	:CONTRol < port >
:DIGital	:POLarity
:DATA < port > ?	:POLarity?
:BIT < number > ?	[:VALue]
:BLOCK?	:DATA < port >
:FLAG < port > ?	[:VALue]
	:BIT < number >
MEMory	:TRACe
:DELete	:HANDshake
MACRO	:DELay
:VME	[:MODE]
:ADDRess	[:MODE]?
:ADDRess?	
:SIZE	:POLarity
:SIZE?	:POLarity?
:STATe	:FLAG < port >
:STATe?	:POLarity
	:POLarity?
	:HANDshake < port >
	:DELay
	[:MODE]
	[:MODE]?

**System Instrument
Commands**

Table A-14. System Instrument SCPI-1990.0 Confirmed Commands

STATus			:PARity
:QUEStionable			[:TYPE]
:CONDition?			[:TYPE]?
[:EVENT]?			:CHECK
:ENABle			:CHECK?
:ENABle?			:SBITS
:OPERation			:SBITS?
:CONDition?			:TRANsmit
[:EVENT]?			:AUTO
:ENABle			:AUTO?
:ENABle?			
:PREset		:ERRor?	
		:TIME?	
		:DATE	
		:DATE?	
		:VERSion?	
SYSTem			
:COMMunicate			
:GPIB			
:ADDRes?		VXI	
:SERial		:CONFigure	
[:RECEive]		:DNUMBER?	
:BAUD			
:BAUD?			
:BITS			
:BITS?			

Table A-15. System Instrument SCPI-1991.0 Confirmed Commands

SYSTem			
:COMMunicate			
SERial			
[:RECEive]			
:PACE			
		[:PROTOcol]	
		[:PROTOcol]?	
		:THReshold	
		:STARt	
		:STARt?	
		:STOP	
		:STOP?	
:TRANsmit			
:PACE			
		[:PROTOcol]	
		[:PROTOcol]?	
:CONTrol			
:RTS			
:RTS?			
:DTR			
:DTR?			

Table A-15. System Instrument SCPI-1992.0 Approved Commands

VXI	VXI	VXI
:SElect	:WSProtocol	:WSProtocol
:CONFigure	:COMMand	:QUERy
:HEIRarchy	:CLOCK	[:ANY?]
:ALL	:CRESpone	:AHLine?
:INFormation	:ENO	:AILine?
:ALL	:GDEvice	:AMControl?
:LADdress?	:ICOMmander	:ANO?
:NUMBer?	:RDEvice	:BNO?
:REGister	:RHANDlers	:BRQ?
:READ?	:RHLine	:CEVent?
:WRITe	:RILine	:CRESpone?
:RESet?	:RINTerrupter	:ENO?
VXI	:RMODid	:RDEvice?
:WSProtocol	:RPERror	:RHANDlers?
:COMMand	:RPRotocol	:RHLine?
[:ANY]	:RSTB	:RILine?
:AHLine	:RSARea	:RINTerrupter?
:AILine	:SLModid	:RMODid?
:AMControl	:SLOCK	:RPERror?
:ANO	:SUModid	:RPRotocol?
:BAVailable	:TRIGger	:RSTB?
:BNO	:MESSAge	:RSARea?
:BRQ	:RECeive?	:SLModid?
:CEVent	:SEND	:SUModid?
:CLR		:RESpense?

Table A-16. System Instrument Non-SCPI Commands

DIAGnostic	:NRAM
:COMMunicate	:CREate
:SERial	:CREate?
[:OWNer]	:AVailable?
[:OWNer]?	:RDISK
:BOOT	:CREate
:COLD	:CREate?
[:WARM]	:ADDress?
:UPLoad?	:PEEK
[:MADDRESS]	:POKE
:SADDRESS	
:DOWNload	MEMory
CHECKed	:DELete
[:MADDRESS]	:MACRo
:SADDRESS	VXI
[:MADDRESS]	:CONFigure
:SADDRESS	:CTABLE
:INTerrupt	:DCTable
:SETup(n)	:DLADdress?
:SETup(n)?	:DEVICELADd?
:PRiority(n)	:DLIST?
:PRiority(n)?	:DEVICELISt?
:WAIT?	:DEVICENUMber?
:DRIVer	:ETABLE
:LOAD	:ITABLE
:LIST?	:READ?
:DRAM	:RECeive[:MESSAge]
:CREate	:RESet
:CREate?	:SEND:COMMand
:AVailable?	:SEND[:MESSAge]
	:WRITe

Table A-17. Common Commands SCPI-1990.0 Confirmed

*IDN	*RCL
*RST	*SAV
*TST	*TRG
*CLS	*DMC
*ESE	*GMC?
*ESE?	*PMC
*ESR	*LMC?
*SRE	*EMC
*SRE?	*EMC?
*STB	*OPC
*PSC	*OPC?
*PSC?	*WAI

Error Messages

Using This Appendix

This appendix shows how to read an instrument's error queue, discusses the types of command language-related error messages, and provides a table of all of the System Instrument's error messages and their probable causes.

- Reading an Instrument's Error QueueB-1
- Error TypesB-2
- Startup Error Messages and WarningsB-3

Reading an Instrument's Error Queue

Executing the SYST:ERR? command reads the oldest error message from the instrument's error queue and erases that error from the error queue. The SYST:ERR? command returns response data in the form:

< error number > , < error description string > .

Example error message; -113,"Undefined header"

Positive error numbers are specific to an instrument. Negative error numbers are command language-related and discussed in the next section "Error Messages". Command language-related errors also set a corresponding bit in the Standard Event Status Register (refer to Chapter 4 for more information).

Example: Reading the Error Queue

This program reads all errors (one error at a time, oldest to newest) from the System Instrument's (Command Module) error queue. After reading each error, that error is automatically erased from the queue. When the error queue is empty, this program returns: +0,"No error".

```

10 OPTION BASE 1
20 DIM Message$(256)           Create array for error message
30 REPEAT                     Repeat next 3 lines until error
                               number = 0
40   OUTPUT 70900;"SYST:ERR?"  Read error number & message
50   ENTER 70900;Code,Message$ Enter error number & message
60   PRINT Code,Message$      Print error number & message
70   UNTIL Code=0
80 END
    
```

Error codes read from the error queue are preceded by the number 21. For example, error code 11 displayed on a monitor appears as 2111 if read from the error queue instead.

Error Types

Negative error numbers are language-related and categorized as shown in Table B-1. Positive error numbers are instrument specific and for the System Instrument are summarized in Table B-2. For other instruments, refer to their own user's manual for a description of error messages.

Table B-1. Negative Error Numbers

Error Number	Error Type
-199 to -100	Command Errors
-299 to -200	Execution Errors
-399 to -300	Device-Specific Errors
-499 to -400	Query Errors

Command Errors

A command error means the instrument cannot understand or execute the command. When a command error occurs, it sets the Command Error Bit (bit 5) in the Standard Event Status Register. Command errors can be caused by:

- A syntax error was detected in a received command or message. Possible errors include a data element which violates the instrument's listening formats or is of the wrong type (binary, numeric, etc.) for the instrument.
- An unrecognizable command header was received. Unrecognizable headers include incorrect SCPI headers and incorrect or unimplemented Common Commands.
- A Group Execute Trigger (GET) was entered into the input buffer inside of a Common Command.

Execution Errors

An execution error indicates the instrument is incapable of doing the action or operation requested by a command. When an execution error occurs, it sets the Execution Error Bit (bit 4) in the Standard Event Status Register. Execution errors can be caused by the following:

- A parameter within a command is outside the limits or inconsistent with the capabilities of an instrument.
- A valid command could not be executed because of an instrument failure or other condition.

Device-Specific Errors

A device-specific error indicates an instrument operation did not complete, possibly due to an abnormal hardware or firmware condition (self-test failure, loss of calibration or configuration memory, etc.). When a device-specific error occurs, it sets the Device-Specific Error Bit (bit 3) in the Standard Event Status Register.

Query Errors

A query error indicates a problem has occurred in the instrument's output queue. When a query error occurs, it sets the Query Error Bit (bit 2) in the Standard Event Status Register. Query errors can be caused by the following:

- An attempt was made to read the instrument's output queue when no output was present or pending.
- Data in the instrument's output queue has been lost for some reason.

Table B-2. Error Messages and Causes

Error Messages and Causes		
Code	Message	Cause
-101	Invalid character	Unrecognized character in specified parameter.
-102	Syntax error	Command is missing a space or comma between parameters
-103	Invalid separator	Command parameter is separated by some character other than a comma.
-104	Data type error	The wrong data type (i.e. number, character, string expression) was used when specifying a parameter.
-108	Parameter not allowed	Parameter specified in a command which does not require one.
-109	Missing parameter	No parameter specified in the command in which a parameter is required.
-113	Undefined header	Command header was incorrectly specified.
-123	Numeric overflow	A parameter specifies a value greater than the command allows.
-128	Numeric data not allowed	A number was specified for a parameter when a letter is required.
-131	Invalid suffix	Parameter suffix incorrectly specified (e.g. .5SECOND rather than .5S or .5SEC).
-138	Suffix not allowed	Parameter suffix is specified when one is not allowed.
-141	Invalid character data	The discrete parameter specified is not allowed (e.g. TRIG:SOUR INT - INT is not a choice.)
-178	Expression data not allowed	A parameter other than the channel list is enclosed in parentheses.
-211	Trigger ignored	Trigger occurred while the Pacer is in the idle state, or a trigger occurred from a source other than the specified source.
-222	Data out of range	The parameter value specified is too large or too small.
-224	Illegal parameter value	The numeric value specified is not allowed.
-240	Hardware error	Hardware error detected during power-on cycle. Return multimeter to Hewlett-Packard for repair.
-310	System error	If caused by *DMC, then macro memory is full.
-350	Too many errors	The error queue is full as more than 30 errors have occurred.
-410	Query interrupted	Data is not read from the output buffer before another command is executed.
-420	Query unterminated	Command which generates data not able to finish executing due to a multimeter configuration error.
-430	Query deadlocked	Command execution cannot continue since the mainframe's command input, and data output buffers are full. Clearing the instrument restores control.
1500	External trigger source already allocated	"Event In" signal already allocated to another instrument such as a Switchbox.
2002	Invalid logical address	A value less than 0 or greater than 255 was specified for logical address.
2003	Invalid word address	An odd address was specified for a 16 bit read or write. Always use even addresses for 16 bit (word) accesses.
2005	No card at logical address	A non-existent logical address was specified with the VXI:READ? or VXI:WRITE command.
2101	Failed Device	VXI device failed its self test.
2102	Unable to combine device	Device type can not be combined into an instrument such as a scanning voltmeter or a switchbox.
2103	Config warning, Device driver not found	ID of device does not match list of drivers available. Warning only.
2105	Config error 5, A24 memory overflow	More A24 memory installed in the mainframe than can be configured into the available A24 memory space.
2108	Config error 8, Inaccessible A24 memory	A24 memory device overlaps memory space reserved by the mainframe's operating system.

Error Messages and Causes		
Code	Message	Cause
2110	Config error 10, Insufficient system memory	Too many instruments installed for the amount of RAM installed in the mainframe. Cannot configure instruments. Only the system instrument is started.
2111	Config error 11, Invalid instrument address	A device's logical address is not a multiple of 8 and the device is not part of a combined instrument.
2112	Invalid user defined commander logical address	The commander assigned to a device by a user defined Configuration Table does not assign it a secondary address.
2114	Invalid user defined secondary address	A secondary address assigned by a user configuration table is illegal
2115	Duplicate secondary address	A secondary address specified by a user configuration table is used more than once.
2116	Invalid servant area	The logical address plus servant area of a commander is greater than 255 or greater than that of a superior commander within this tree.
2117	Slot 0 functions disabled	A command module is in slot 0 but slot 0 switches are in the disabled position.
2118	Invalid commander logical address	A device does not have a valid commander
2119	BNO failed	Sending a BEGIN Normal Operation command to a device failed.
2120	Write ready timeout	A message based device failed to become write ready.
2121	Read ready timeout	A message based device failed to become read ready.
2122	ERR* asserted	The ERR* bit is asserted in a device's response register.
2123	ENO failed	Sending an End Normal Operation command to a device failed
2124	Interrupt line unavailable	no line is available for a programmable interrupt handler. All lines are used or duplicate.
2125	Invalid user defined handler	The user defined interrupt table specifies a device that is not a programmable interrupt handler, or does not exist.
2126	Invalid user defined interrupter	The user defined interrupt table specifies a device that is not a programmable interrupter, or does not exist.
2127	Diagnostic mode on	HP-IB address switch bit 6 is set wrong (warning only).
2128	Resource Manager not in Slot 0	A Command Module is configured for Slot 0 and Resource Manager but is installed in another slot (warning only).
2129	Warning, Sysfail detected	A device was asserting SYSFAIL on the backplane during startup.
2130	Pseudo instrument logical address unavailable	A physical device has the same logical address as IBASIC (240)
2131	File system start up failed	Insufficient system resources to allow the IBASIC file system to start.
2133	Invalid UDEF memory block	Invalid memory block in user defined Memory table.
2134	UDEF memory block unavailable	The same base address or memory are specified more than once in the Memory table, or the addresses in the specified block are already in use.
2135	Invalid UDEF address space	The address specified in the Memory table is A24 but the device is A32, or vice versa.
2136	Duplicate UDEF memory LADD	A logical address is specified more than once in the Memory table. This does not apply to VME devices (address = -1).
2137	Invalid UDEF CNFG table	The valid flag in the Command/Servant Heirarchy table is not set to 1.
2138	Invalid UDEF CNFG table data	There are more than 254 entries in the Commander/Servant Heirarchy table.
2139	Invalid UDEF DC table	The valid flag in the Dynamic Configuration table is not set to 1.
2140	Invalid UDEF DC table data	There are more than 254 entries in the Dynamic Configuration Table.

Error Messages and Causes		
Code	Message	Cause
2141	Invalid UDEF Interrupter	The logical address specified for an interrupter is a device that is not an interrupter.
2142	Invalid UDEF INTR table	The Interrupter table valid flag is not 1.
2143	Invalid UDEF MEM table	The valid flag in the Memory table is not set to 1.
2144	Invalid UDEF MEM table data	An invalid logical address is specified in the Memory table.
2145	Warning, Non-volatile RAM contents lost	NVRAM was corrupted or a cold boot was executed.
2146	MESG based open access failed	I or I4 device is violating VXI specification
2147	Granted device not found	
2148	Warning, DRAM contents lost	Driver RAM was corrupted or a cold boot was executed.
2149	VME system controller disabled	VME SYSTEM CONTROLLER switch is disabled on the E1405 module.
2150	Extender not slot 0 device	VXIbus extender in remote mainframe is not in slot 0 of its mainframe.
2151	Invalid extender LADD window	MXI extender cannot be configured with a valid LADD window.
2152	Device outside of LADD window	A device is located outside the allowable logical address window range of an MXIbus extender.
2153	Invalid extender A24 window	MXIbus extender cannot be configured with a valid A24 memory window.
2154	Device outside of A24 window	An A24 memory device is located outside the allowable logical address window range of an MXIbus extender.
2155	Invalid extender A32 window	MXIbus extender cannot be configured with a valid A32 memory window.
2156	Device outside of A32 window	An A32 memory device is located outside the allowable logical address window range of an MXIbus extender.
2157	Invalid UDEF LADD window	User defined logical address window has incorrect base address or size.
2158	Invalid UDEF A16 window	User defined A16 memory window has incorrect base address or size.
2159	Invalid UDEF A24 window	User defined A24 memory window has incorrect base address or size.
2160	Invalid UDEF A32 window	User defined A32 memory window has incorrect base address or size.
2161	Invalid UDEF EXT table	The valid flag in the Extender table is not set to 1
2162	Invalid UDEF extender table data	There are more than 254 records in the Extender table.
2163	Unsupported UDEF TTL trigger	There is an extender table TTL trigger entry for a device which does not support TTL triggers.
2164	Unsupported UDEF ECL trigger	There is an extender table ECL trigger entry for a device which does not support ECL triggers.
2165	Device not in configure state	A message based device was not in CONFIGURE state during re-boot.
2166	INTX card not installed	The INTX daughter card on the VXI-MXI module is not installed or is not functioning correctly.
2201	Unexpected interrupt from message based card	A message based card interrupted when an interrupt service routine has not been set up.
2202	Unexpected interrupt from non-message based card	A register based card interrupted when an interrupt service routine had not been set up.
2809	Interrupt line has not been set up	A DIAG:INT:ACT or DIAG:INT:RESP command was executed before setting the interrupt with DIAG:INT:SET.
2810	Not a handler for this line	An attempt was made to set up an interrupt with DIAG:INT:SET for a line that has no handler. (see VXI:CONF:ITAB).

Start-up Error Messages and Warnings

Start-up error messages and warnings are most often generated just after the mainframe is powered-up or re-booted (DIAG:BOOT command). These messages can be read from the error queue using the SYST:ERR? command. We recommend that you include a routine at the beginning of your application programs which checks for start-up errors before the program tries to access individual instruments. See your Installation and Getting Started Guide for an example program.

Table B-3. Start-up Error Messages and Warnings

Start-Up Error Messages and Warnings		
Code	Message	Cause
1	Failed Device	VXI device failed its self test.
2	Unable to combine device	Device type can not be combined into an instrument such as a scanning voltmeter or a switchbox.
3	Config warning, Device driver not found	ID of device does not match list of drivers available. Warning only.
4	DC device block too big	Dynamically configured device address block is greater than 127.
5	Config error 5, A24 memory overflow	More A24 memory is installed in the mainframe than can be configured into the available A24 memory space.
6	A32 memory overflow	More A32 memory is installed in the mainframe than can be configured into the available A32 memory space.
7	DC device move failed	A dynamically configured device failed to move to a new logical address.
8	Config error 8, Inaccessible A24 memory	An A24 memory device overlaps a memory space reserved by the mainframe's operating system.
9	Unable to move DC device	The block size for a set of address-blocked Dynamically Configured devices is too large for the available space or an attempt was made to move a Dynamically Configured device to an already assigned Logical Address. Cannot configure instruments. Only the system instrument is started.
10	Config error 10, Insufficient system memory	Too many instruments installed for the amount of RAM installed in the mainframe. Cannot configure instruments. Only the system instrument is started.
11	Config error 11, Invalid instrument address	A device's logical address is not a multiple of 8 and the device is not part of a combined instrument.
12	Invalid user defined commander logical address	The commander assigned to a device by a user defined Configuration Table does not assign it a secondary address.
14	Invalid user defined secondary address	A secondary address assigned by a user configuration table is illegal
15	Duplicate secondary address	A secondary address specified by a user configuration table is used more than once.
16	Invalid servant area	The logical address plus servant area of a commander is greater than 255 or greater than that of a superior commander within this tree.
17	Slot 0 functions disabled	A command module is in slot 0 but slot 0 switches are in the disabled position.
18	Invalid commander logical address	A device does not have a valid commander
19	BNO failed	Sending a BEGIN Normal Operation command to a device failed.
20	Write ready timeout	A message based device failed to become write ready.
21	Read ready timeout	A message based device failed to become read ready.
22	ERR* asserted	The ERR* bit is asserted in a device's response register.

Start-Up Error Messages and Warnings		
Code	Message	Cause
23	ENO failed	Sending an End Normal Operation command to a device failed
24	Interrupt line unavailable	no line is available for a programmable interrupt handler. All lines are used or duplicate.
25	Invalid user defined handler	The user defined interrupt table specifies a device that is not a programmable interrupt handler, or does not exist.
26	Invalid user defined interrupter	The user defined interrupt table specifies a device that is not a programmable interrupter, or does not exist.
27	Diagnostic mode on	HP-IB address switch bit 6 is set wrong (warning only).
28	Resource Manager not in Slot 0	A Command Module is configured for Slot 0 and Resource Manager but is installed in another slot (warning only).
29	Warning, Sysfail detected	A device was asserting SYSFAIL on the backplane during startup.
30	Pseudo instrument logical address unavailable	A physical device has the same logical address as IBASIC (240)
31	File system startup failed	Insufficient system resources to allow the IBASIC file system to start.
32	Inaccessible A32 memory	Device has A32 memory below 200000000 ₁₆ or above DFFFFFFF ₁₆
33	Invalid UDEF memory block	Invalid memory block in user defined Memory table.
34	UDEF memory block unavailable	The same base address or memory are specified more than once in the Memory table, or the addresses in the specified block are already in use.
35	Invalid UDEF address space	The address specified in the Memory table is A24 but the device is A32, or vice versa.
36	Duplicate UDEF memory LADD	A logical address is specified more than once in the Memory table. This does not apply to VME devices (address = -1).
37	Invalid UDEF CNFG table	The valid flag in the Command/Servant Heirarchy table is not set to 1.
38	Invalid UDEF CNFG table data	There are more than 254 entries in the Commander/Servant Heirarchy table.
39	Invalid UDEF DC table	The valid flag in the Dynamic Configuration table is not set to 1.
40	Invalid UDEF DC table data	There are more than 254 entries in the Dynamic Configuration Table.
41	Invalid UDEF Interrupter	The logical address specified for an interrupter is a device that is not an interrupter.
42	Invalid UDEF INTR table	The Interrupter table valid flag is not 1.
43	Invalid UDEF MEM table	The valid flag in the Memory table is not set to 1.
44	Invalid UDEF MEM table data	An invalid logical address is specified in the Memory table.
45	Warning, NVRAM contents lost	NVRAM was corrupted or a cold boot was executed.
46	MESG based open access failed	I or I4 device is violating VXI specification
47	Granted device not found	
48	Warning, DRAM contents lost	Driver RAM was corrupted or a cold boot was executed.
49	VME system controller disabled	VME SYSTEM CONTROLLER switch is disabled on the E1405 module.
50	Extender not slot 0 device	VXIbus extender in remote mainframe is not in slot 0 of its mainframe.
51	Invalid extender LADD window	MXI extender cannot be configured with a valid LADD window.
52	Device outside of LADD window	A device is located outside the allowable logical address window range of an MXIbus extender.
53	Invalid extender A24 window	MXIbus extender cannot be configured with a valid A24 memory window.
54	Device outside of A24 window	An A24 memory device is located outside the allowable logical address window range of an MXIbus extender.
55	Invalid extender A32 window	MXIbus extender cannot be configured with a valid A32 memory window.

Start-Up Error Messages and Warnings		
Code	Message	Cause
56	Device outside of A32 window	An A32 memory device is located outside the allowable logical address window range of an MXibus extender.
57	Invalid UDEF LADD window	User defined logical address window has incorrect base address or size.
58	Invalid UDEF A16 window	User defined A16 memory window has incorrect base address or size.
59	Invalid UDEF A24 window	User defined A24 memory window has incorrect base address or size.
60	Invalid UDEF A32 window	User defined A32 memory window has incorrect base address or size.
61	Invalid UDEF EXT table	The valid flag in the Extender table is not set to 1
62	Invalid UDEF extender table data	There are more than 254 records in the Extender table.
63	Unsupported UDEF TTL trigger	There is an extender table TTL trigger entry for a device which does not support TTL triggers.
64	Unsupported UDEF ECL trigger	There is an extender table ECL trigger entry for a device which does not support ECL triggers.
65	Device not in configure state	A message based device was not in CONFIGURE state during re-boot.
66	INTX card not installed	The INTX daughter card on the VXI-MXI module is not installed or is not functioning correctly.

Command Module A16 Address Space

About this Appendix

Many Hewlett-Packard VXIbus devices are register-based devices which do not support the VXIbus word serial protocol. When an SCPI command is sent to a register-based device, the HP E1405 Command Module parses the command and programs the device at the register level.

Register-based programming is a series of reads and writes directly to the device registers. This increases throughput since it eliminates command parsing.

This appendix contains an address map of A16 address space in the Command Module. It shows how to determine the base address and register offset for register-based devices mapped into A16 space. Refer to the individual plug-in module manuals for details on device is programming at the register level.

Register Addressing

Register addresses for register-based devices are located in the upper 25% of VXI A16 address space. Every VXI device (up to 256 devices per Command Module) is allocated a 64 byte block of addresses. A device may or may not use the entire block of addresses. Figure C-1 shows the location of A16 address space in the HP E1405 Command Module.

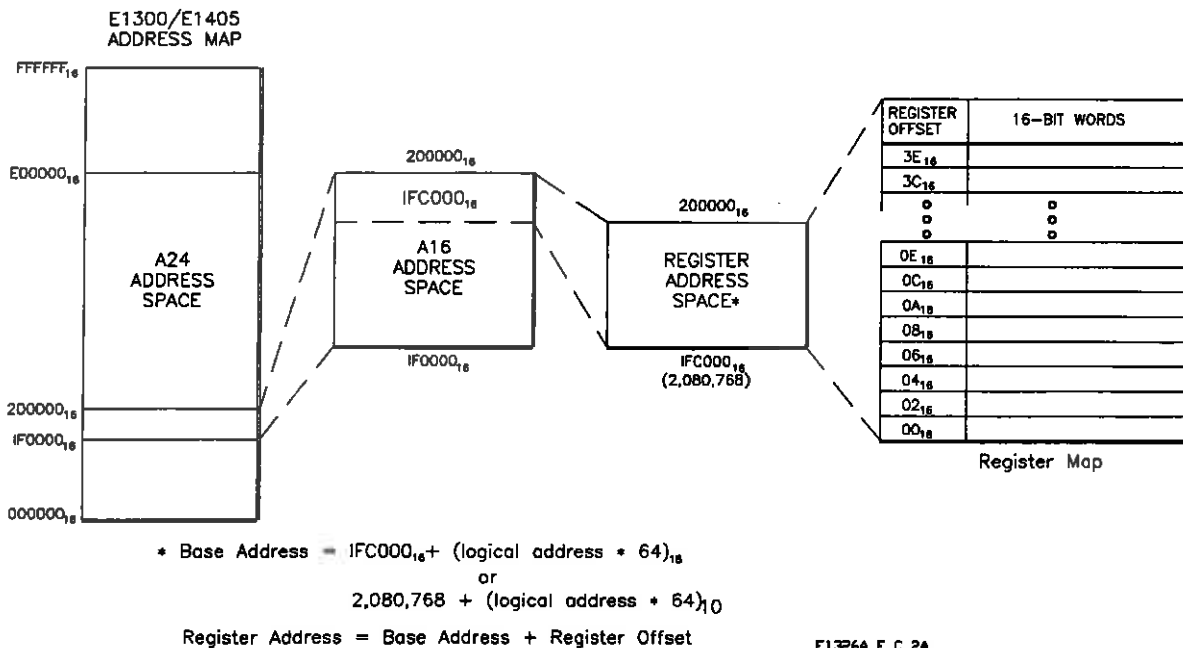


Figure C-1. HP E1405 Command Module A16 Address Space

The Base Address

When you are reading or writing to a device register, a hexadecimal or decimal register address is specified. This address consists of a base address plus a register offset.

Determining the Base Address

The base address of a device in A16 address space is computed as:

$$1FC000_{16} + (LADDR * 64)_{16}$$

or

$$2,080,768_{10} + (LADDR * 64)_{10}$$

where $1FC000_h$ ($2,080,768_{10}$) is the starting location of the VXI A16 addresses, LADDR is the device's logical address, and 64 is the number of address bytes per register-based device. For example, the HP E1411 multimeter has a factory set logical address of 24. If this address is not changed, the multimeter will have a base address of:

$$1FC000_{16} + (24 * 64)_{16}$$
$$1FC000_{16} + 600_{16} = 1FC600_{16}$$

or

$$2,080,768_{10} + (24 * 64)_{10}$$
$$2,080,768_{10} + 1536_{10} = 2,082,304_{10}$$

Register Offset

The register offset is the register's location in the block of 64 address bytes. For example, the HP E1411 multimeter's Command Register has an offset of 08h. When you write a command to this register, the offset is added to the base address to form the register address:

$$1FC600_{16} + 08_{16} = 1FC608_{16}$$

or

$$2,082,304_{10} + 8_{10} = 2,082,312_{10}$$

Sending Binary Data Over RS-232

About this Appendix

This appendix describes the procedure for sending pure binary data over an RS-232 interface. The formatting described is used in the DIAG:DOWN:CHEC:MADD, DIAG:DOWN:CHEC:SADD, and DIAG:DRIV:LOAD:CHEC commands. this appendix contains the following main sections.

- About this Appendix D-1
- Formatting Binary Data for RS-232 Transmission D-1
- Sending Binary Data Over RS-232 D-2

Formatting Binary Data for RS-232 Transmission

The most straightforward way to send a block of data is to open the data file, read the next byte from the file, and send it to the System Instrument until you reach the end of file. However, binary data cannot be sent to the system instrument as is. It must be converted into a format that will not conflict with the special characters that the RS-232 interface recognizes. This is done by sending only one half byte (a nibble) at a time. To prevent this nibble from being confused with a special character, bit 7 of the nibble is set to one. This gives all data bytes in the block values greater than 127 so they are not confused with ASCII characters. It also doubles the size of the file to be sent and the transmission time for the file. Since a transmission error that required retransmission of the entire data block would be very time consuming, a 3-bit error code (which allows for correction of single bit errors) is added to the transmission byte. The following format is sent for each nibble:

Bit #	7	6	5	4	3	2	1	0
	1	<i>Correction Code</i>			<i>Data</i>			

The error correction code is based on the nibble of data sent. The easiest way to implement this code is to use table D-1. It is indexed based on the value of the nibble to send out, so there are 16 elements to the table.

Table D-1. Correction Codes for RS-232 Transmission

Data Value	Correction Code	Byte in Hex	Byte in Decimal
0	0	80 ₁₆	128
1	7	F1 ₁₆	241
2	6	E2 ₁₆	226
3	1	93 ₁₆	147
4	5	D4 ₁₆	212
5	2	A5 ₁₆	165
6	3	B6 ₁₆	182
7	4	C7 ₁₆	199
8	3	B8 ₁₆	184
9	4	C9 ₁₆	201
10	5	DA ₁₆	218
11	2	AB ₁₆	171
12	6	EC ₁₆	236
13	1	9D ₁₆	157
14	0	8E ₁₆	142
15	7	FF ₁₆	255

Sending Binary Data Over RS-232

The RS-232 interface differs from the HP-IB interface in that there is no device addressing built into the interface definition. Device addressing must be done on top of the RS-232 functions. This addressing is done through the same mechanism as the terminal-based front panel, and must be done either by the transfer program or manually before starting the transfer program.

Setting Up the Mainframe

There are two commands (SI - Select and Instrument and SA - Select Address) that can be used at the "Select an Instrument" interface. The "Select an Instrument" interface can always be reached by sending the < CTRL-D > character (ASCII 4) over the RS-232 line. Once there, the System Instrument can be reached by sending the command "SI SYSTEM" followed by a carriage return. All output after this command will be directed to/from the System Instrument until another < CTRL-D > is received. The following sequence will make sure that the mainframe is set up and ready.

1. Send < CTRL-D > (ASCII 4) to get to the "Select and Instrument" interface.
2. Send "ST UNKNOWN" and a carriage return to insure that the interface is set to dumb terminal mode.

3. Send "SI SYSTEM" and a carriage return to get the attention of the System Instrument.
4. Send < CTRL-C > to clear the system.
5. Send "**RST" and a carriage return to put the System Instrument in a known state.

The program must then send the binary data. This block of data should include the command "DIAG:DOWN:CHEC" followed by the address to download to and an IEEE 488.2 arbitrary block header. This block header can be either definite or indefinite. The advantage of using an indefinite block header is that you do not need to know the length of the data block. The indefinite block header is #0. With the DIAG:DOWN:CHEC command an indefinite block is terminated with the "!" character followed by a carriage return. The "!" character is not considered part of the block. A definite block only requires the ASCII carriage return character as terminator. The definite block starts with #. This is followed by a single digit that shows the number of digits in the length field, which is followed by the actual length of the block, not counting the header. For instance, a block of 1000 bytes would have a definite block header of #41000. Due to the formatting required, the size of the block when using the DIAG:DOWN:CHEC command is twice the length of the data in bytes.

Once the block header has been sent, the actual data is sent. Since the buffer size of the System Instrument RS-232 Interface is limited to 79 bytes, the buffer must be flushed (passed to an instrument parser) before it reaches 79 bytes. This can be done by sending a carriage return. The first carriage return should be included in the binary file after the buffer header. Sending it before this would result in the parser determining that there are not enough parameters and producing an error condition. Once transmission of the actual data begins, a carriage return should be included after every 78 bytes.

NOTE

The carriage returns are not considered part of the block count.

After the last byte of data, there must be a carriage return to terminate the transmission for a definite block or a "!" and carriage return for an indefinite block.

Index

!

- 3-Channel Universal Counter
 - menu (terminal interface), 3-43 - 3-44
- 4-Channel Counter/Totalizer
 - menu (terminal interface), 3-41 - 3-42
- 4-Channel D/A Converter
 - menu (terminal interface), 3-38
- 5 1/2 Digit Multimeter
 - menu (terminal interface), 3-37
- 60-second menu tutorial
 - terminal interface, 3-3

A

- A24/A32 Address Allocation table, 2-26
 - data format, 2-27
 - example, 2-28
 - table format, 2-26
 - table size, 2-27
- A24/A32 memory
 - address allocation, 2-22
 - address mapping, 2-22
 - reserving address space, 2-26
- Abbreviated Commands, 6-2
- Address allocation
 - A24/A32, 2-22
- Address mapping
 - A24/A32, 2-22
- Address space for VME devices, 2-23
- Address window configuration, 2-8
- Addressing a register, C-1
- Annunciators
 - faceplate, 1-2

B

- Back Space key
 - terminal interface, 3-19
- Base address
 - A16 address space, C-2
- Battery backed functions, 1-3
- BOOT
 - :COLD, 6-5
 - :WARM, 6-5

C

- Cable
 - RS-232, 5-1
- Caps Lock key
 - terminal interface, 3-19
- Clear-to-end key
 - terminal interface, 3-19
- Clearing Standard Operation Event Register Bits, 4-9
- Clearing status, 4-10
- CLK10 connectors
 - faceplate, 1-2
- Close channels
 - terminal interface, 3-12
- Command
 - Abbreviated, 6-2
 - Implied, 6-2
 - Linking, 6-3
 - Separator, 6-2
 - Types, 6-1
- Command Errors, B-2
- Command Module
 - installing/removing, 1-3
- Command module electrical description, 1-2
- Command module functional description, 1-1
- Command module physical description, 1-2
- Command Quick Reference, 6-93
- Command Reference, SCPI
 - Common Commands, 6-83
 - DIAGnostic subsystem, 6-4, 6-6 - 6-26
 - OUTPut subsystem, 6-27 - 6-31, 6-33 - 6-34
 - STATus subsystem, 6-35 - 6-37
 - SYSTEM subsystem, 6-38 - 6-53
 - VXI subsystem, 6-54 - 6-82
- Commander/servant hierarchies
 - setting, 2-17
 - user-defined, 2-18
- Commander/Servant Hierarchy Table, 2-18
 - data format, 2-19
 - example, 2-20
 - table format, 2-18
 - table size, 2-19
- Commands
 - executing (terminal interface), 3-17
 - terminal interface, 3-24
- Common Command Format, 6-1
- Common Command reference, 6-83
- Common Command reference, all instruments
 - *CLS, 6-84
 - *ESE, 6-84
 - *ESE?, 6-85

- *ESR?, 6-85
- *IDN?, 6-86
- *LRN?, 6-86
- *OPC, 6-87
- *OPC?, 6-87
- *PSC, 6-87
- *PSC?, 6-87
- *RST, 6-88
- *SRE, 6-88
- *SRE?, 6-88
- *STB?, 6-89
- *TST?, 6-89
- *WAI, 6-89
- Common Commands functional groupings, 6-83
- COMMunicate:GPIB
 - :ADDRess?, 6-39
- COMMunicate:SERial[0]
 - :OWNer, 6-6
 - :OWNer?, 6-6
- COMMunicate:SERial[n]
 - :CONTRol:DTR, 6-40
 - :CONTRol:DTR?, 6-40
 - :CONTRol:RTS, 6-41
 - :CONTRol:RTS?, 6-41
 - :RECEive:BAUD, 6-42
 - :RECEive:BAUD?, 6-42
 - :RECEive:BITS, 6-43
 - :RECEive:BITS?, 6-43
 - :RECEive:PACE:PROTocol, 6-44
 - :RECEive:PACE:PROTocol?, 6-44
 - :RECEive:PACE:THReshold:STARt, 6-45
 - :RECEive:PACE:THReshold:STARt?, 6-45
 - :RECEive:PACE:THReshold:STOP, 6-46
 - :RECEive:PACE:THReshold:STOP?, 6-46
 - :RECEive:PARity:CHECK, 6-47
 - :RECEive:PARity:CHECK?, 6-47
 - :RECEive:PARity:TYPE, 6-47 - 6-48
 - :RECEive:PARity:TYPE?, 6-49
 - :RECEive:SBITs, 6-49
 - :RECEive:SBITs?, 6-50
 - :STORE, 6-7
 - :TRANsmit:AUTO, 6-50
 - :TRANsmit:AUTO?, 6-50
 - :TRANsmit:PACE:PROTocol, 6-51
 - :TRANsmit:PACE:PROTocol?, 6-51
- Condition register, reading, 4-8
- Configuration
 - A24/A32 address allocation, 2-22
 - A24/A32 Address Allocation table, 2-26
 - A24/A32 address mapping, 2-22
 - address space for VME devices, 2-23
 - address window, 2-8
 - Command Module RS-232 port, 3-11
 - commander/servant hierarchies, 2-17
 - dynamically configured modules, 2-2
 - ECL trigger register, 2-10
 - interrupt line allocation, 2-30
 - interrupt register, 2-9

- logical address, 2-7
- statically configured modules, 2-2
- system configuration sequence, 2-1
- TTL trigger register, 2-9
- user defined Dynamic configuration, 2-2
- utility register, 2-10
- CONFigure
 - :CTABle, 6-55
 - :CTABle?, 6-56
 - :DCTable, 6-56
 - :DCTable?, 6-57
 - :DLADdress?, 6-57
 - :DLISt?, 6-58
 - :DNUMber?, 6-59
 - :HIERarchy:ALL?, 6-61
 - :HIERarchy?, 6-60
 - :INFormation:ALL?, 6-62
 - :INFormation?, 6-61
 - :ITABle, 6-63
 - :ITABle?, 6-63
 - :LADdress:MEXTender?, 6-64
 - :MEXTender:ECLTrg, 6-64
 - :MEXTender:INTerrupt, 6-65
 - :MEXTender:TTLtrg, 6-66
 - :MTABle, 6-67
 - :MTABle?, 6-67
 - :NUMBer:MEXTender?, 6-68
 - :NUMBer?, 6-64, 6-68
- Connectors
 - CLK10 and trigger faceplate, 1-2
- Control keys, menu (terminal interface), 3-18

D

- Data format
 - A24/A32 Address Allocation table, 2-27
 - Commander/Servant Hierarchy table, 2-19
 - Interrupt Line Allocation table, 2-34
 - user defined DC table, 2-4
 - user defined Extender table, 2-12
- DATE, 6-51
- DATE?, 6-52
- DCL (device clear), 6-91
- Delete key
 - terminal interface, 3-18
- Device clear (DCL), 6-91
- Device Driver
 - manual download over HP-IB, 5-11
 - manual download over RS-232, 5-11
 - preparing memory for download, 5-10
- Device Driver RAM, 5-3
- Device Drivers
 - checking status, 5-9
 - Disks, 5-1
 - downloading in HP-IB systems with HP BASIC, 5-8
 - downloading in HP-IB systems with IBASIC, 5-7
 - downloading in MS-DOS systems, 5-6
 - downloading multiple drivers, 5-9

- editing the configuration file, 5-4
- Memory Configuration, 5-3
- Device-Specific Errors, B-2
- DIAGnostic subsystem, 6-4, 6-6 - 6-26
 - DIAG:BOOT:COLD, 6-5
 - DIAG:BOOT:WARM, 6-5
 - DIAG:COMM:SER[0]:OWN, 6-6
 - DIAG:COMM:SER[0]:OWN?, 6-6
 - DIAG:COMM:SER[n]:STOR, 6-7
 - DIAG:DOWN:CHEC:SADD, 6-10 - 6-11
 - DIAG:DOWN:CHEC[:MADD], 6-8 - 6-9
 - DIAG:DOWN:SADD, 6-13
 - DIAG:DOWN[:MADD], 6-12
 - DIAG:DRAM:AVA?, 6-14
 - DIAG:DRAM:CRE, 6-14 - 6-15
 - DIAG:DRIV:LIST:ROM?, 6-17
 - DIAG:DRIV:LIST?, 6-16
 - DIAG:DRIV:LOAD:CHEC, 6-16
 - DIAG:DRIV:LOAD, 6-15
 - DIAG:INT:ACT, 6-17
 - DIAG:INT:PRI[n], 6-19
 - DIAG:INT:PRI[n]?, 6-20
 - DIAG:INT:RESP?, 6-20
 - DIAG:INT:SET[n], 6-18
 - DIAG:INT:SET[n]?, 6-18
 - DIAG:NRAM:ADDR?, 6-21
 - DIAG:NRAM:CRE, 6-21
 - DIAG:NRAM:CRE?, 6-22
 - DIAG:PEEK?, 6-22
 - DIAG:POKE, 6-23
 - DIAG:RDIS:ADD?, 6-23
 - DIAG:RDIS:CRE, 6-24
 - DIAG:RDIS:CRE?, 6-24
 - DIAG:UPL:SADD?, 6-26
 - DIAG:UPL[:MADD]?, 6-25
 - DRIV:LIST:RAM?, 6-16
- Display
 - annunciators, monitor mode, 3-16
 - module type and description (terminal interface), 3-12
- DOWNload
 - :CHECked:SADDress, 6-10 - 6-11
 - :CHECked[:MADDress], 6-8 - 6-9
 - :SADDress, 6-13
 - [:MADDress], 6-12
- Downloading device drivers
 - checking status, 5-9
 - hardware handshake, 5-12
 - in HP-IB systems with HP BASIC, 5-8
 - in HP-IB systems with IBASIC, 5-7
 - in MS-DOS systems, 5-6
 - manually over HP-IB, 5-11
 - manually over RS-232, 5-11
 - manually using hardware handshake, 5-13
 - manually using software handshake, 5-14
 - multiple device drivers, 5-9
 - pacing data, 5-12
 - preparing memory, 5-10
 - software handshake, 5-12

- DRAM, 5-3
 - :AVAlable?, 6-14
 - :CREate, 6-14
 - :CREate?, 6-15
- DRIVer
 - :LIST?, 6-16
 - :LOAD:CHECked, 6-16
 - LIST:RAM?, 6-16
 - LIST:ROM?, 6-17
- DRIVer:LOAD, 6-15
- Drivers
 - listing, 6-16
- Dynamic configuration
 - user-defined, 2-2
- Dynamic Configuration table, 2-3
 - data format, 2-4
 - example, 2-5
 - table format, 2-3
 - table size, 2-4
- Dynamically configured modules, 2-2

E

- ECL trigger register configuration, 2-10
- ECLTrg
 - :IMMediate, 6-27
 - :LEVel[:IMMediate], 6-28
 - :LEVel[:IMMediate]?, 6-28
 - :SOURce, 6-28
 - :SOURce?, 6-29
 - [:STATe], 6-29
 - [:STATe]?, 6-30
- Editing
 - keys (terminal interface), 3-18
 - the configuration file, 5-4
 - VXIDLD.CFG, 5-4
- electrical description
 - Command Module, 1-2
- Error
 - messages, reading, 3-16
 - SYST:ERR?, 6-52
- Error Messages, B-1 - B-8
- Error queue, reading, B-1
- Error Types, B-2
- ERRor?, 6-52
- Errors
 - Command, B-2
 - Device-Specific, B-2
 - Execution, B-2
 - Query, B-2
- Example
 - Assigning a secondary HP-IB address, 2-20
 - Assigning an interrupt line, 2-35
 - dynamically configuring a module, 2-5
 - interrupting when an error occurs, 4-11
 - reading the error queue, B-1
 - Reserving addresses for a VME device, 2-28
 - synchronizing computers using *OPC, 4-13

- synchronizing computers using *OPC?, 4-12
- user defined Extender table, 2-13
- Using the Operation Status Group Registers, 4-9
- Executing commands (terminal interface), 3-17
- Execution Errors, B-2
- Extender table, 2-10
 - data format, 2-12
 - determining size, 2-12
 - example, 2-13
 - table format, 2-11
- EXTERNAL
 - :IMMEDIATE, 6-30
 - :LEVEL[:IMMEDIATE], 6-30
 - :LEVEL[:IMMEDIATE]?, 6-31
 - :SOURCE, 6-31
 - :SOURCE?, 6-31
 - [:STATE], 6-32
 - [:STATE]?, 6-32
- External computer, interrupting, 4-10
- External computer/instruments, synchronizing, 4-12

F

- Faceplate annunciators, 1-2
- Faceplate CLK10 and trigger connectors, 1-2
- Files
 - VXIDLD.CFG, 5-4
- Format
 - Common Command, 6-1
 - SCPI Command, 6-1
- functional description
 - command module, 1-1
- Functional groupings, Common Commands, 6-83

G

- GET (group execute trigger), 6-90
- Go to local (GTL), 6-90
- Group execute trigger (GET), 6-90
- GTL (go to local), 6-90

H

- How instruments appear in the menu, 3-3
- HP-IB message reference, 6-90

I

- IFC (interface clear), 6-90
- Implied Commands, 6-2
- In case of difficulty
 - terminal interface, 3-27
- Installing/removing the Command Module, 1-3
- Instrument
 - Control Keys (terminal interface), 3-19
 - menus (terminal interface), 3-29
 - menus, using, 3-12

- Interface clear (IFC), 6-90
- INTERRUPT
 - :ACTivate, 6-17
 - :PRiority[n], 6-19
 - :PRiority[n]?, 6-20
 - :RESPonse?, 6-20
 - :SETup[n], 6-18
 - :SETup[n]?, 6-18
- Interrupt Line Allocation, 2-30
 - user defined, 2-32
- Interrupt Line Allocation table, 2-32
 - data format, 2-34
 - example, 2-35
 - table format, 2-32
 - table size, 2-33
- Interrupt register configuration, 2-9
- Interrupting external computer, 4-10

K

- Key descriptions, General, 3-18
- Keys
 - editing (terminal interface), 3-18
 - menu (terminal interface), 3-18
 - menu control (terminal interface), 3-18

L

- Left arrow key
 - terminal interface, 3-18
- Linking Commands, 6-3
- LLO (local lockout), 6-91
- Local lockout (LLO), 6-91
- Locating statically and dynamically configured modules, 2-2
- Logical address configuration, 2-7

M

- Memory
 - Device Driver RAM, 5-3
- Menu
 - control keys (terminal interface), 3-18
 - how instruments appear, 3-3
 - instrument (terminal interface), 3-29
 - keys (terminal interface), 3-18
 - multiple Command Modules, 3-3
 - tutorial (terminal interface), 3-3
 - using a terminal without, 3-25
- Menu (terminal interface)
 - 3-Channel Universal Counter, 3-43 - 3-44
 - 4-Channel Counter/Totalizer, 3-41 - 3-42
 - 4-Channel D/A Converter, 3-38
 - 5 1/2 Digit Multimeter, 3-37
 - Quad 8-Bit Digital Input/Output, 3-39
 - Scanning Voltmeter, 3-35 - 3-36
 - Switchbox, 3-33

- System Instrument, 3-30 - 3-32
- Mode, monitor, 3-15
- Modules
 - dynamically configured, 2-2
 - statically configured, 2-2
- Monitor
 - a Switchbox (terminal interface), 3-12
 - mode, 3-15
 - mode, display annunciators, 3-16
- Multiple Command Modules, 3-3
- Multiple device drivers, 5-9

N

- NRAM, 5-5
 - :ADDRess?, 6-21
 - :CREate, 6-21
 - :CREate?, 6-22

O

- Open and close channels
 - terminal interface, 3-12
- OPERation
 - :CONDition?, 6-35
 - :ENABle, 6-36
 - :ENABle?, 6-36
 - [:EVENT]?, 6-37
- Other Terminals, non-supported, 3-24
- OUTPut subsystem, 6-27 - 6-31, 6-33 - 6-34
 - OUTP:ECLT:IMM, 6-27
 - OUTP:ECLT:LEV[:IMM], 6-28
 - OUTP:ECLT:LEV[:IMM]?, 6-28
 - OUTP:ECLT:SOUR, 6-28
 - OUTP:ECLT:SOUR?, 6-29
 - OUTP:ECLT[:STAT], 6-29
 - OUTP:ECLT[:STAT]?, 6-30
 - OUTP:EXT:IMM, 6-30
 - OUTP:EXT:SOUR, 6-31
 - OUTP:EXT:SOUR?, 6-31
 - OUTP:EXT[:STATe], 6-32
 - OUTP:EXT[:STATe]?, 6-32
 - OUTP:LEV[:IMM], 6-30
 - OUTP:LEV[:IMM]?, 6-31
 - OUTP:TTLT, 6-32
 - OUTP:TTLT:LEV[:IMM], 6-33
 - OUTP:TTLT:LEV[:IMM]?, 6-33
 - OUTP:TTLT:SOUR, 6-33
 - OUTP:TTLT:SOUR?, 6-34
 - OUTP:TTLT[:STAT], 6-34
 - OUTP:TTLT[:STAT]?, 6-35

P

- Pacing data for manual download, 5-12
- PEEK?, 6-22
- Physical description
 - Command Module, 1-2
- POKE, 6-23
- PRESet, 6-37

Q

- Quad 8-Bit Digital Input/Output
 - menu (terminal interface), 3-39
- Query Errors, B-2
- QUERy?, 6-68
- QUESTionable, 6-37 - 6-38
- Quick Reference, Command, 6-93

R

- RDISK, 5-5
 - :ADDRess?, 6-23
 - :CREate, 6-24
 - :CREate?, 6-24
- READ?, 6-69
- Reading
 - an instrument's error queue, B-1
 - error messages (terminal interface), 3-16
 - the Condition register, 4-8
 - the Status Byte register, 4-4
- Reference, Common Commands, 6-83
- REGister
 - :READ?, 6-71
 - :WRITe, 6-72
- Register addressing, C-1
- Register offset, C-2
- Register, Status Byte, 4-4
- Remote (HP-IB message), 6-92
- Removing
 - /installing the Command Module, 1-3
- Reserving A24/A32 address space, 2-26
- Reserving A24/A32 addresses, 2-26
- Reset, 6-73
 - a switch module (terminal interface), 3-12
- Reset button, 1-3
- RESet?, 6-73
- Right arrow key
 - terminal interface, 3-18
- ROUTE
 - :ECLTrg, 6-74
 - :INTerrupt, 6-74
 - :TTLTrg, 6-75
- RS-232 Cable, 5-1

S

- SA, terminal interface command, 3-25
- Scan channels
 - Switchbox, terminal interface, 3-12
- Scanning Voltmeter
 - menu (terminal interface), 3-35 - 3-36
- SCPI Commands, 6-1
 - Format, 6-1
 - Reference, 6-4
- SDC (selected device clear), 6-91
- SELEct, 6-75
- Select Address command (terminal interface), 3-25
- Select Instrument command (terminal interface), 3-25
- SELEct?, 6-75
- Selected device clear (SDC), 6-91
- Selecting
 - instruments, without menus, 3-25
 - the Switchbox (terminal interface), 3-12
- SEND
 - :COMMand, 6-76
 - :COMMand?, 6-77
 - [:MESSAge], 6-78
- Separator
 - Command, 6-2
- Serial poll (SPOLL), 6-92
- Service request
 - enable register, 4-5
- Service request enable register, 4-5
 - clearing, 4-5
- Setting Commander/servant hierarchies, 2-17
- Shift key
 - terminal interface, 3-19
- SI, terminal interface command, 3-25
- SPOLL (serial poll), 6-92
- ST, terminal interface command, 3-24
- Standard Commands for Programmable Instruments, SCPI, 6-4
- Standard Event Status bits, unmasking, 4-6
- Standard Event Status Register, 4-6
 - reading, 4-7
- Standard Event Status Register (table), 4-6
- Standard operation status group
 - condition register, 4-7
- Standard Operation Status Group - Condition register (table), 4-8
- Start system operation, 2-37
- Statically and dynamically configured modules
 - locating, 2-2
- Statically configured modules, 2-2
- Status Byte register, 4-3 - 4-4
- Status Byte Register, reading, 4-4
- STATus subsystem, 6-35 - 6-37
 - STAT:OPER:COND?, 6-35
 - STAT:OPER:ENAB, 6-36
 - STAT:OPER:ENAB?, 6-36
 - STAT:OPER[:EVEN]?, 6-37
 - STAT:PRES, 6-37
 - STAT:QUES, 6-37 - 6-38
- Status system structure, 4-2
- Status, clearing, 4-10
- Status, system structure, 4-2
- Subsystem
 - DIAGnostic, 6-4, 6-6 - 6-26
 - OUTput, 6-27 - 6-31, 6-33 - 6-34
 - STATus, 6-35 - 6-37
 - SYSTem, 6-38 - 6-53
 - VXI, 6-54 - 6-82
- Switchbox
 - close channels (terminal interface), 3-12
 - menu (terminal interface), 3-33
 - monitoring (terminal interface), 3-12
 - open and close channels (terminal interface), 3-12
 - scan channels (terminal interface), 3-12
 - selecting (terminal interface), 3-12
 - show module data (terminal interface), 3-12
- Synchronizing external computer/instruments, 4-12
- Syntax, Variable Command, 6-2
- System Configuration Sequence, 2-1
- System Instrument, 6-1
 - menu (terminal interface), 3-30 - 3-32
- System Instrument menu, 3-5
- SYSTem subsystem, 6-38 - 6-53
 - SYST:COMM:GPIB:ADDR?, 6-39
 - SYST:COMM:SER[n]:REC:PAR:TYPE, 6-47 - 6-48
 - SYST:COMM:SER[n]:CONT:DTR, 6-40
 - SYST:COMM:SER[n]:CONT:DTR?, 6-40
 - SYST:COMM:SER[n]:CONT:RTS?, 6-41
 - SYST:COMM:SER[n]:REC:BAUD, 6-42
 - SYST:COMM:SER[n]:REC:BAUD?, 6-42
 - SYST:COMM:SER[n]:REC:BITS, 6-43
 - SYST:COMM:SER[n]:REC:BITS?, 6-43
 - SYST:COMM:SER[n]:REC:PACE:PROT, 6-44
 - SYST:COMM:SER[n]:REC:PACE:PROT?, 6-44
 - SYST:COMM:SER[n]:REC:PACE:THR:STAR, 6-45
 - SYST:COMM:SER[n]:REC:PACE:THR:STAR?, 6-45
 - SYST:COMM:SER[n]:REC:PACE:THR:STOP, 6-46
 - SYST:COMM:SER[n]:REC:PACE:THR:STOP?, 6-46
 - SYST:COMM:SER[n]:REC:PAR:CHEC, 6-47
 - SYST:COMM:SER[n]:REC:PAR:CHEC?, 6-47
 - SYST:COMM:SER[n]:REC:PAR:TYPE?, 6-49
 - SYST:COMM:SER[n]:REC:SBIT, 6-49
 - SYST:COMM:SER[n]:REC:SBIT?, 6-50
 - SYST:COMM:SER[n]:TRAN:AUTO, 6-50
 - SYST:COMM:SER[n]:TRAN:AUTO?, 6-50
 - SYST:COMM:SER[n]:TRAN:PACE:PROT, 6-51
 - SYST:COMM:SER[n]:TRAN:PACE:PROT?, 6-51
 - SYST:COMM:SERial[n]:CONT:RTS, 6-41
 - SYST:DATE, 6-51
 - SYST:DATE?, 6-52
 - SYST:ERRor?, 6-52
 - SYST:TIME, 6-53
 - SYST:TIME?, 6-53
 - SYST:VERS?, 6-53 - 6-54

T

Table format

- A24/A32 Address Allocation table, 2-26
- Commander/Servant Hierarchy table, 2-18
- Interrupt Line Allocation table, 2-32
- user defined DC table, 2-3
- user defined Extender table, 2-11

Table size

- A24/A32 Address Allocation table, 2-27
- Commander/Servant Hierarchy table, 2-19
- Interrupt Line Allocation table, 2-33
- user defined DC table, 2-4
- user defined Extender table, 2-12

Tables

- Commander/Servant Hierarchy table, 2-18
- Interrupt Line Allocation table, 2-32
- user defined DC table, 2-3
- user defined Extender table, 2-10

Terminal interface

- commands, 3-24
- commands, SA, 3-25
- commands, SI, 3-25
- commands, ST, 3-24
- features, 3-1
- menu tutorial, 3-3
- menus, 3-2

TIME, 6-53

TIME?, 6-53

Trigger connectors

- faceplate, 1-2

TTL trigger register configuration, 2-9

TTLTrg

- :IMMediate, 6-32
- :LEVel[:IMMediate], 6-33
- :LEVel[:IMMediate]?, 6-33
- :SOURce, 6-33
- :SOURce?, 6-34
- [:STATe], 6-34
- [:STATe]?, 6-35

U

Unmasking Standard Event Status bits, 4-6

Unmasking Standard Operation Event Register Bits, 4-8

UPLoad

- :SADDress?, 6-26
- [:MADDress]?, 6-25

User defined tables

- A24/A32 Address Allocation table, 2-26
- Commander/Servant Hierarchy table, 2-18
- Dynamic Configuration table, 2-2
- Extender table, 2-10
- Interrupt Line Allocation table, 2-32

Using

- a terminal without menus, 3-25

instrument menus (terminal interface), 3-12

menus, 3-2

Operation Status Group Registers, 4-9

Other Terminals, 3-24

Supported Terminals, 3-20

Utility register configuration, 2-10

V

Variable Command Syntax, 6-2

VERSion?, 6-53 - 6-54

VME devices

address space, 2-23

reserving A24 address space, 2-26

reserving A24/A32 addresses, 2-26

VXI subsystem, 6-54 - 6-82

VXI:CONF:CTAB, 6-55

VXI:CONF:CTAB?, 6-56

VXI:CONF:DCT, 6-56

VXI:CONF:DCT?, 6-57

VXI:CONF:DLAD?, 6-57

VXI:CONF:DLIS?, 6-58

VXI:CONF:DNUM?, 6-59

VXI:CONF:HIER:ALL?, 6-61

VXI:CONF:HIER?, 6-60

VXI:CONF:INF:ALL?, 6-62

VXI:CONF:INF?, 6-61

VXI:CONF:ITAB, 6-63

VXI:CONF:ITAB?, 6-63

VXI:CONF:LADD:MEXT?, 6-64

VXI:CONF:MEXT:ECLT, 6-64

VXI:CONF:MEXT:INT, 6-65

VXI:CONF:MEXT:TTLT, 6-66

VXI:CONF:MTAB, 6-67

VXI:CONF:MTAB?, 6-67

VXI:CONF:NUMB:MEXT?, 6-68

VXI:CONF:NUMB?, 6-64, 6-68

VXI:QUER?, 6-68

VXI:READ?, 6-69

VXI:REG:READ?, 6-71

VXI:REG:WRITE, 6-72

VXI:RES, 6-73

VXI:RES?, 6-73

VXI:ROUT:ECLT, 6-74

VXI:ROUT:INT, 6-74

VXI:ROUT:TTLT, 6-75

VXI:SEL, 6-75

VXI:SEL?, 6-75

VXI:SEND:COMM, 6-76

VXI:SEND:COMM?, 6-77

VXI:SEND[:MESS], 6-78

VXI:WRIT, 6-79

VXI:WSP:COMM. . ., 6-80

VXI:WSP:MESS:REC?, 6-81

VXI:WSP:MESS:SEND, 6-81

VXI:WSP:QUER. . .?, 6-82

VXI:WSP:RESP?, 6-82

VXI-MXI configuration, 2-7
VXIDLD.CFG, 5-4

W

WRITE, 6-79

WSProtocol

:COMMand. . ., 6-80

:MESSAge:RECeive?, 6-81

:MESSAge:SEND, 6-81

:QUERy. . .?, 6-82

WSProtocol:RESPonse?, 6-82

SALES & SUPPORT OFFICES

Arranged alphabetically by country



HEADQUARTERS OFFICES

If there is no sales office listed for your area, contact one of these headquarters offices.

NORTH/CENTRAL AFRICA

Hewlett-Packard S.A.
7, rue du Bois-du-Lan
CH-1217 MEYRIN 1, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hmea
Cable: HEWPACKSA Geneve

ASIA

Hewlett-Packard Asia Ltd.
47/F, 26 Harbour Rd.,
Wanchai, HONG KONG
G.P.O. Box 863, Hong Kong
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HPASIAL TD

EASTERN EUROPE

Hewlett-Packard Ges.m.b.h.
Liebigasse 1
P.O.Box 72
A-1222 VIENNA, Austria
Tel: (222) 2500-0
Telex: 13 4425 HEPA A

NORTHERN EUROPE

Hewlett-Packard S.A.
V. D. Hooplaan 241
P.O.Box 999
NL-1183 AG AMSTELVEEN
The Netherlands
Tel: 20 547999
Telex: 18 919 hpner

SOUTH EAST EUROPE

Hewlett-Packard S.A.
World Trade Center
110 Avenue Louis Casal
1215 Cointrin, GENEVA, Switzerland
Tel: (022) 98 96 51
Telex: 27225 hpser

EASTERN USA

Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370

MIDWESTERN USA

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA

Hewlett-Packard Co.
2000 South Park Place
ATLANTA, GA 30339
Tel: (404) 955-1500

WESTERN USA

Hewlett-Packard Co.
5161 Lankershim Blvd.
NORTH HOLLYWOOD, CA 91601
Tel: (818) 505-5600

MEDITERRANEAN AND MIDDLE EAST

Hewlett-Packard S.A.
Mediterranean and Middle East
Operations
Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion, ATHENS
Greece
Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

OTHER INTERNATIONAL AREAS

Hewlett-Packard Co.
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK

ARGENTINA

Hewlett-Packard Argentina S.A.
Montaneses 2140/50
1428 BUENOS AIRES
Tel: 781-4059/69
Cable: HEWPACKARG

AUSTRALIA

Hewlett-Packard Australia Ltd.
31-41 Joseph Street
P.O. Box 221
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 888-4444
Telex: 21561
Cable: HEWPARD Sydney

AUSTRIA

Hewlett-Packard Ges.m.b.h.
Liebigasse 1
P.O. Box 72
A-1222 VIENNA
Tel: (0222) 2500-0
Telex: 134425 HEPA A

BELGIUM

Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 BRUSSELS
Tel: (02) 762-32-00
Telex: 23-494 paloben bru

BRAZIL

Hewlett-Packard do Brasil
I.e.C. Ltda.
Alameda Rio Negro, 750
ALPHAVILLE
06400 Barueri SP
Tel: (011) 421.1311
Telex: (011) 33872 HPBR-BR
Cable: HEWPACK Sao Paulo

Hewlett-Packard do Brasil
I.e.C. Ltda.
Praia de Botafogo 228
6° Andar-conj 614

Edificio Argentina - Ala A
22250 RIO DE JANEIRO, RJ
Tel: (021) 552-6422
Telex: 21905 HPBR-BR
Cable: HEWPACK Rio de Janeiro

CANADA

Hewlett-Packard (Canada) Ltd.
11120-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666

Hewlett-Packard (Canada) Ltd.
17500 Trans Canada Highway
South Service Road
KIRKLAND, Quebec H9J 2X8
Tel: (514) 697-4232
Telex: 058-21521

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 069-8644

Hewlett-Packard (Canada) Ltd.
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483

CHINA, People's Republic of

China Hewlett-Packard Co., Ltd.
P.O. Box 9610, Beijing
4th Floor, 2nd Watch Factory Main Bldg.
Shuang Yu Shou, Bei San Huan Road
Hai Dian District
BEIJING
Tel: 28-0567
Telex: 22601 CTSHP CN
Cable: 1920 Beijing

DENMARK

Hewlett-Packard A/S
Kongevejen 25
DK-3460 BIRKERØD
Tel: (02) 81-66-40
Telex: 37409 hpas dk

FINLAND

Hewlett-Packard Oy
Pilsankallontie 17
02200 ESPOO
Tel: 00358-0-88721
Telex: 121583 HEWPA SF

FRANCE

Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
69131 ECULLY Cedex (Lyon)
Tel: (78) 133-81-25
Telex: 310617F

Hewlett-Packard France
Parc d'activités du Bois Briard
Avenue du Lac
91040 EVRY Cedex
Tel: (60) 77-83-83
Telex: 602315F

Hewlett-Packard France
Zone Industrielle de Courtaboeuf
Avenue des Tropiques
91947 LES ULIS Cedex (Orsay)
Tel: (69) 07-78-25
Telex: 600048F

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Vertriebszentrum Mitte
Hewlett-Packard-Strasse
D-6380 BAD HOMBURG
Tel: (06172) 400-0
Telex: 410 844 hpbhg

Hewlett-Packard GmbH
Vertriebszentrum Südwest
Schickardstrasse 2
D-7030 BÖBLINGEN
Tel: (07031) 645-0
Telex: 7265 743 hep

Hewlett-Packard GmbH
Vertriebszentrum Süd
Eschenstrasse 5
D-8028 TAUFKIRCHEN
Tel: (089) 61 20 7-0
Telex: 0524985

GREECE

Hewlett-Packard A.E.
178, Kifissias Avenue
6th Floor
Halendri-ATHENS
Greece
Tel: 6471543, 6471673, 6472971
Telex: 221 286 HPHLGR

HONG KONG

Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPACK HONG KONG

ICELAND

Hewlett-Packard Iceland
Hoefdabakka 9
110 REYKJAVIK
Tel: (1) 67 1000

INDIA

Blue Star Ltd.
13 Community Center
New Friends Colony
NEW DELHI 110 065
Tel: 633182, 636674
Telex: 031-61120
Cable: BLUEFROST

INDONESIA

BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 11th Floor
Jl. Medan Merdeka Selatan 17
JAKARTA-PUSAT
Tel: 343989
Telex: 46748 BERSAL IA

IRELAND

Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2
Tel: 0001 608800
Telex: 30439

ISRAEL

Computation and Measurement
Systems (CMS) Ltd.
11 Masad Street
67060
TEL-AVIV
Tel: 388 388
Telex: 33569 Motil IL

ITALY

Hewlett-Packard Italiana S.p.A.
Via G. di Vittorio 9
I-20063 CERNUSCO SUL
NAVIGLIO
(Milano)
Tel: (02) 923691
Telex: 334632

Hewlett-Packard Italiana S.p.A.
Viale C. Pavese 340
I-00144 ROMA EUR
Tel: (06) 54831
Telex: 610514

JAPAN

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624
Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku TOKYO 168
Tel: (03) 331-6111
Telex: 232-2024 YHPTOK

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
Kanagawa-ku, YOKOHAMA 221
Tel: (045) 312-1252

SALES & SUPPORT OFFICES

Arranged alphabetically by country

**KOREA**

Samsung Hewlett-Packard Co. Ltd.
Dongbang Yeouuido Building
12-16th Floors
36-1 Yeouuido-Dong
Youngdeungpo-Ku
SEOUL
Tel: 784-4666, 784-2666
Telex: 25166 SAMSAN K

MALAYSIA

Hewlett-Packard Sales (Malaysia)
Sdn. Bhd.
9th Floor
Chung Khtew Bank Building
46, Jalan Raja Laut
50350 KUALA LUMPUR
Tel: 2986555
Telex: 31011 HPSM MA

MEXICO

Hewlett-Packard de Mexico,
S.A. de C.V.
Monte Pelvoux No. 111
Lomas de Chapultepec
11000 MEXICO, D.F.
Tel: 5-40-82-28, 72-88, 50-25
Telex: 17-74-507 HEWPACK MEX.

NETHERLANDS

Hewlett-Packard Nederland B.V.
Startbaan 16
NL-1187 XR AMSTELVEEN
P.O. Box 667
NL-1180 AR AMSTELVEEN
Tel: (020) 547-8911
Telex: 13 216 HEPA NL

NORWAY

Hewlett-Packard Norge A/S
Ostermdalen 16-18
P.O. Box 34
N-1345 OESTERAAS
Tel: 0047/2/24 60 90
Telex: 76621 hpnas n

PUERTO RICO

Hewlett-Packard Puerto Rico
101 Muñoz Rivera Av
Esu. Calle Ochoa
HATO REY, Puerto Rico 00918
Tel: (809) 754-7800

SAUDI ARABIA

Modern Electronics Establishment
Hewlett-Packard Division
P.O. Box 1228
Redec Plaza, 6th Floor
JEDDAH
Tel: 644 96 28
Telex: 4027 12 FARNAS SJ
Cable: ELECTA JEDDAH

SINGAPORE

Hewlett-Packard Singapore (Sales)
Pte. Ltd.
#08-00 Inchcape House
450-2 Alexandra Road
Alexandra P.O. Box 58
SINGAPORE, 91115
Tel: 4731788
Telex: 34209 HPSGSO RS
Cable: HEWPACK, Singapore

SOUTH AFRICA

Hewlett-Packard So Africa (Pty.) Ltd.
9 Eastern Service Road
Eastgate Ext. 3
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877 SA
Cable: HEWPACK Johannesburg

SPAIN

Hewlett-Packard Española, S.A.
Ctra. de la Coruña, Km. 18, 400
Las Rozas
E-MADRID
Tel: (1) 837.00.11
Telex: 23515 HPE

SWEDEN

Hewlett-Packard Sverige AB
Skalhögsgatan 9, Kista
Box 19
S-16393 SPÅNGA
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781

SWITZERLAND

Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case postale 365
CH-1217 MEYRIN 1
Tel: (0041) 22-83-11-11
Telex: 27333 HPAG CH

TAIWAN

Hewlett-Packard Taiwan Ltd.
8th Floor, Hewlett-Packard Building
337 Fu Hsing North Road
TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable: HEWPACK Taipei

TURKEY

Teknim Company Ltd.
Iran Caddesi No. 7
Karaklidere
ANKARA
Tel: 275800
Telex: 42155 TKNM TR

UNITED KINGDOM

ENGLAND
Hewlett-Packard Ltd.
Heathside Park Road
Cheadle Heath
STOCKPORT
Cheshire
SK3 ORB
Tel: 061-428-0828
Telex: 668068

Hewlett-Packard Ltd.
King Street Lane
Winnersh, WOKINGHAM
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178

SCOTLAND

Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682

UNITED STATES

Alabama
Hewlett-Packard Co.
420 Wynn Drive
HUNTSVILLE, AL 35805
Tel: (205) 830-2000

Arizona
Hewlett-Packard Co.
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000

California
Hewlett-Packard Co.
1421 S. Manhattan Av.
FULLERTON, CA 92631
Tel: (714) 998-8700

Hewlett-Packard Co.
5651 West Manchester Ave.
LOS ANGELES, CA 90045
Tel: (213) 337-8000
Telex: 910-325-6608

Hewlett-Packard Co.
9606 Aero Drive
SAN DIEGO, CA 92123
Tel: (619) 279-3200

Hewlett-Packard Co.
3003 Scott Boulevard
SANTA CLARA, CA 95054
Tel: (408) 988-7000
Telex: 910-338-0586

Colorado

Hewlett-Packard Co.
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 648-5000

Connecticut

Hewlett-Packard Co.
47 Barnes Industrial Road South
WALLINGFORD, CT 06492
Tel: (203) 265-7801

Florida

Hewlett-Packard Co.
2901 N.W. 62nd Street
FORT LAUDERDALE, FL 33309
Tel: (305) 973-2600

Hewlett-Packard Co.
6177 Lake Ellenor Drive
ORLANDO, FL 32809
Tel: (305) 859-2900

Georgia

Hewlett-Packard Co.
2000 South Park Place
ATLANTA, GA 30339
Tel: (404) 955-1500
Telex: 810-766-4890

Illinois

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066

Indiana

Hewlett-Packard Co.
11911 N. Meridian St.
CARMEL, IN 46032
Tel: (317) 844-4100

Louisiana

Hewlett-Packard Co.
160 James Drive East
ST. ROBE, LA 70067
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100

Maryland

Hewlett-Packard Co.
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943

Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370

Massachusetts

Hewlett-Packard Co.
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500

Michigan

Hewlett-Packard Co.
39550 Orchard Hill Place Drive
NOVI, MI 48050
Tel: (313) 349-9200

Minnesota

Hewlett-Packard Co.
2025 W. Larpenteur Ave.
ST. PAUL, MN 55113
Tel: (612) 644-1100

Missouri

Hewlett-Packard Co.
1001 E. 101st Terrace Suite 120
KANSAS CITY, MO 64131-3368
Tel: (816) 941-0411

Hewlett-Packard Co.
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel: (314) 344-5100

New Jersey

Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07653
Tel: (201) 265-5000

New Mexico

Hewlett-Packard Co.
7801 Jefferson N.E.
ALBUQUERQUE, NM 87109
Tel: (505) 823-6100

New York

Hewlett-Packard Co.
9600 Main Street
CLARENCE, NY 14031
Tel: (716) 759-8621

Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820

Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 682-7800

North Carolina

Hewlett-Packard Co.
5805 Roanne Way
GREENSBORO, NC 27420
Tel: (919) 852-1800

Ohio

Hewlett-Packard Co.
15885 Sprague Road
CLEVELAND, OH 44136
Tel: (216) 243-7300

Hewlett-Packard Co.
9080 Springboro Pike
MIAMSBURG, OH 45342
Tel: (513) 433-2223

Hewlett-Packard Co.
675 Brookside Blvd.
WESTERVILLE, OH 43081
Tel: (614) 891-3344

Oklahoma

Hewlett-Packard Co.
2 Choke Cherry Road
3525 N.W. 56th St.
Suite C-100
OKLAHOMA CITY, OK 73112
Tel: (405) 948-9499

Oregon

Hewlett-Packard Co.
9255 S. W. Pioneer Court
WILSONVILLE, OR 97070
Tel: (503) 682-8000

Pennsylvania

Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400

Hewlett-Packard Co.
2750 Monroe Boulevard
VALLEY FORGE, PA 19482
Tel: (215) 666-9000

Texas

Hewlett-Packard Co.
1826-P Kramer Lane
AUSTIN, TX 78758
Tel: (512) 835-8771

Hewlett-Packard Co.
10535 Harwin Drive
HOUSTON, TX 77038
Tel: (713) 778-6400

Hewlett-Packard Co.
930 E. Campbell Rd.
RICHARDSON, TX 75081
Tel: (214) 231-6101

Hewlett-Packard Co.
1020 Central Parkway South
SAN ANTONIO, TX 78232
Tel: (512) 494-9336

Utah

Hewlett-Packard Co.
3530 W. 2100 South St.
SALT LAKE CITY, UT 84119
Tel: (801) 974-1700

Virginia

Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
Tel: (804) 747-7750

Washington

Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000

Wisconsin

Hewlett-Packard Co.
275 N. Corporate Dr.
BROOKFIELD, WI 53005
Tel: (414) 794-8800

VENEZUELA

Hewlett-Packard de Venezuela C.A.
3A Transversal Los Ruices Norte
Edificio Segre 2 & 3
Apartado 50933
CARACAS 1050
Tel: (582) 239-4133
Telex: 251046 HEWPACK

YUGOSLAVIA

Do Hermes
General Zdanova 4
YU-11000 BEOGRAD
Tel: (011) 342 641
Telex: 11433