HP E2749

Fibre Channel Interface Module

User's Guide

HP Part Number E2749-90000

Printed in U.S.A
Print Date: July 1997

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

WARRANTY

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Use of this manual is restricted to this product only.

TRADEMARKS

FibreXpress™ is a trademark of Systran Corporation.

Tachyon™ is a trademark of Hewlett-Packard Company.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure is subject to HP standard commercial license terms or to the following restrictions whichever is applicable:

1) for non-DoD Departments and Agencies of the U.S. Government, as set forth in FAR 52.227-19(c)(1-2)(Jun 1987);

2) for the DoD Departments and its Agencies, as set forth in DFARS 252.227-7013(c)(1)(ii) (Oct 1988), DFARS 252.211-7015(c)(May 1991), whichever is applicable.

HEWLETT-PACKARD COMPANY
3000 Hanover St.
Palo Alto, CA 94304 U.S.A.

Registration Card

Make sure you send in the product registration card located in the red brochure included in your shipping carton. This assures that you will hear about future product and service updates. If the brochure has been misplaced, simply mail or fax the following information to the indicated address/fax number.

- Your Name:
- Position:
- Company Name:
- Division:
- Mail Stop:
- Street:
- City:
- State or Province:
- Postal Code:
- Country:
- Telephone: ( ) -
- Model:
- Serial Number:

Mail to:

ATTN: Customer Support MS 500
Hewlett-Packard Company
Lake Stevens Instrument Division
8600 Soper Hill Rd.
Everett, WA 98205-1298

Or FAX:

(206)335-2828
No cover sheet is required.
Customer Support MS 500

In This Book

The HP E2749 Fibre-Channel Interface Module provides high-speed data transfer between a VXI mainframe and other devices. The module plugs into one C-size slot in a VXI mainframe.

This book documents the HP E2749 module. It provides:

- installation and service information
- operational information
- information on using Sequences with the HP E2749
- SCPI command reference materials
- Information on using the HP E2749A with SYSTRAN Corporation's FibreXpress™ products

TABLE OF CONTENTS

1

Installing the HP E2749

Installing the HP E2749

This chapter contains instructions for installing, transporting, and storing the HP E2749 Fibre Channel Interface Module. For information on verifying module operation, see chapter 2, “Verifying the HP E2749.”

To inspect the HP E2749

The HP E2749 Fibre Channel Interface Module was carefully inspected both mechanically and electrically before shipment. It should be free of marks or scratches, and it should meet its published specifications upon receipt.

If the module was damaged in transit, do the following:

- Save all packing materials.
- File a claim with the carrier.
- Call your Hewlett-Packard sales and service office.

What you get with the HP E2749

The following items are included with your HP E2749:

Standard HP E2749A:

- HP E2749A Fibre Channel Interface Module – C-size 1-slot VXI module, coaxial copper interface, 9-pin D-type connector
- HP E2749 User's Guide (this book)

HP E2749A Option 001:

- HP E2749A Fibre Channel Interface Module – C-size 1-slot VXI module, optical interface, duplex SC connectors

Note

This module includes a Class 1 Laser Product which complies with 21 Code of Federal Regulations (USA, CFR), chapter 1, subchapter J.

- HP E2749 User's Guide (this book).

Option 010:

- 2-meter optical cable

Option 011:

- 2-meter coaxial copper cable with 9-pin D-type connector

Option 0B1:

- additional HP E2749 User's Guide

Accessories available from other suppliers:

- Longer Fibre Channel cables are available from:

  AMP, 800-522-6752
  or
  W.L. Gore & Associates, 800-531-6064

  The following length restrictions apply:

  Optical cables up to 500 m (850 nm multimode laser, 50µm core diameter)
  Coaxial copper cables up to 30 m (150Ω STP connector)

- Compatible Fibre Channel products are available from Systran Corporation, 800-252-5601. Systran Corporation also has a Web site at `www.systran.com` (For more information on Systran products, see the chapter "Using the HP E2749A with Systran FibreXpress Products.")

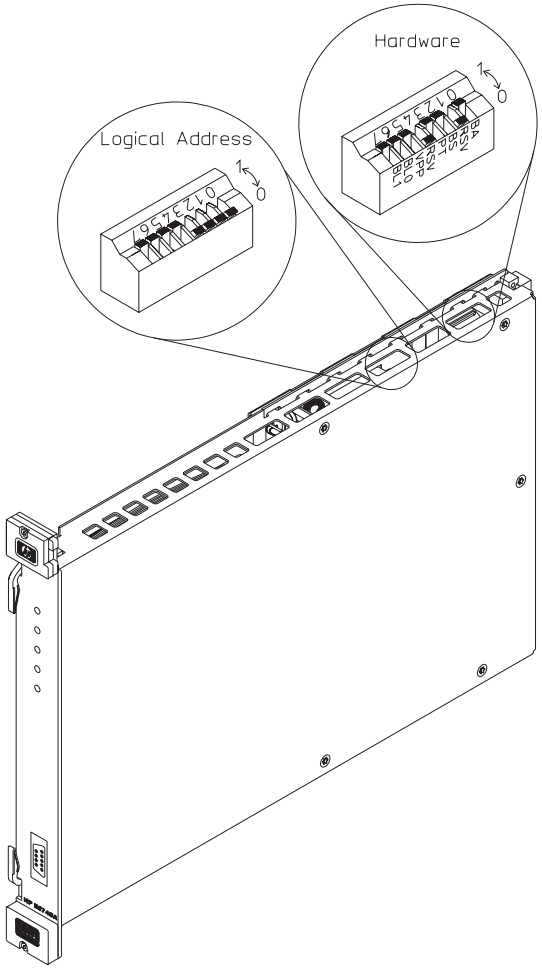
To install the HP E2749

Caution

To protect circuits from static discharge, observe anti-static techniques whenever handling the HP E2749 Fibre Channel Interface Module.

The HP E2749 Fibre Channel Interface can be used in a wide variety of VXI system configurations. The VXI system may use an embedded controller (PC or UNIX), or an external controller connected to the VXI system using an HP-IB, VXLink, or MXIbus interface. Your application program controlling the system may be written in HP-VEE, HP-BASIC, C, Visual Basic, or some other language.

To verify the installation of the HP E2749 module, you must send the "*TST?" query command to the module and read the module's response. Refer to the documentation provided with your application programming language for information on sending commands and reading responses from the module.



1 Set up your VXI mainframe. See the installation guide for your mainframe.

2 Select a slot in the VXI mainframe for the HP E2749 module.

The HP E2749 module receives local bus ECL-level data from the module immediately to its left. If you will be transmitting data from other modules, place the HP E2749 immediately to the right of those modules. If you will be receiving data to be used by other modules, place the HP E2749 immediately to the left of those modules.

3 Using a small screwdriver or similar tool, set the Logical Address configuration switch on the HP E2749.

(See the illustration on previous page.) Each module in the system must have a unique logical address. The factory default setting is 1111 0000 (240). If an HP E1485A Signal Processor Module will be controlling the HP E2749 module, select an address within the HP E1485A module's servant area. If an HP-IB command module will be controlling the HP E2749 module, select an address that is a multiple of 8.

4 Using a small screwdriver or similar tool, set the switches in the Hardware switchpack to 11101101.

(See the illustration on previous page.) The firmware uses switch BA to configure the Fibre Channel data buffers. If the HP E2749A will be primarily transmitting data, set switch BA to ON (1) to allocate more memory for output buffers and less memory to input buffers. Conversely, if the HP E2749A will be primarily receiving data, set switch BA to OFF (0) to allocate more memory for receive buffers.

The BS (boot source) switch determines the behavior of the module's firmware at power-up. BS must be in the ON (normal) position for proper operation.

VPP enables the programming power supplies to programmable ROMs in the module. VPP must be in the ON (service) position to set the module's Fibre Channel WWN. See the chapter titled "SCPI Command Reference" for more information (SYST:COMM:WWN[:SELF]:ADDR command).

RSV (reserved) switches must be set to 0.

Buffer Allocation Optimization	
	BA
Transmit	1
Receive	0

Power-On Test	
	PT
Long	1
Short	0

Bus Request Level		
	BL0	BL1
Level 0	0	0
Level 1	0	1
Level 2	1	0
Level 3	1	1

Boot Source	
	BS
Normal	1
Service	0

ROM Programming Enable	
	VPP
Service	1
Normal	0

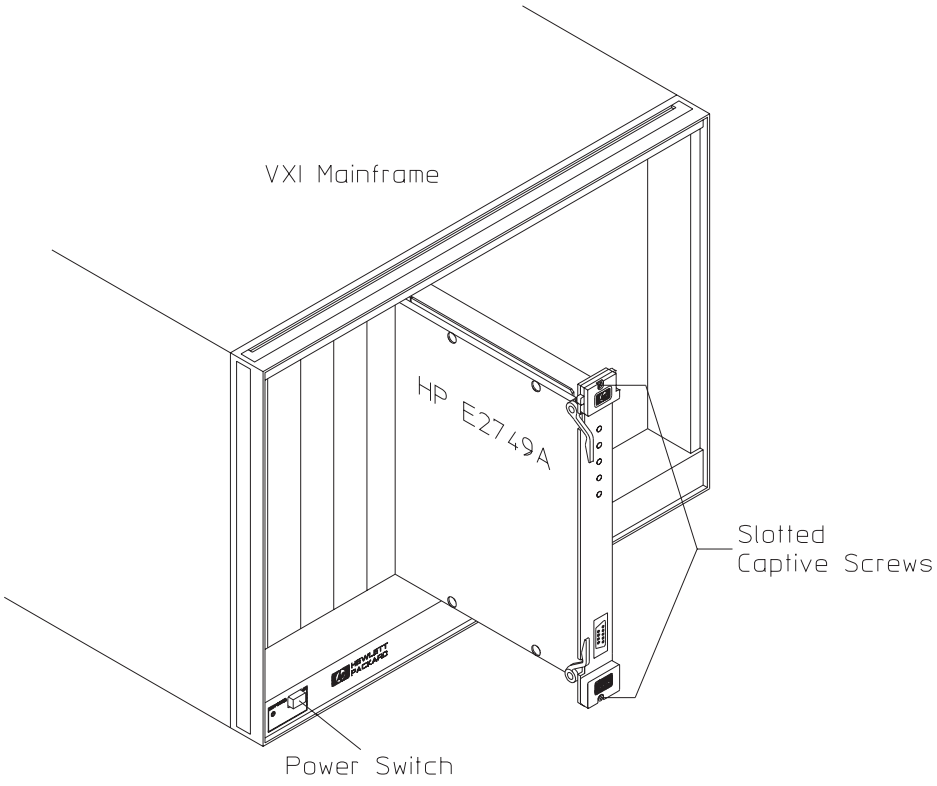
1 Set the mainframe's power switch to standby or off(⏼).

Caution Installing or removing the module with the power on may damage components in the module.

2 Place the module's card edges (top and bottom) into the module guides in the slot.

3 Slide the module into the mainframe until the module connects firmly with the backplane connectors. Make sure the module slides in straight.

4 Attach the module's front panel to the mainframe chassis using the module's captive mounting screws.



To store the module

Store the module in a clean, dry, and static free environment.

For other requirements, see storage and transport restrictions in the document: *HP E2749 Fibre Channel Interface Module Technical Specifications*.

To transport the module

Package the module using the original factory packaging or packaging identical to the factory packaging.
Containers and materials identical to those used in factory packaging are available through Hewlett-Packard offices.

If returning the module to Hewlett-Packard for service, attach a tag describing the following:

- Type of service required
- Return address
- Model number
- Full serial number

In any correspondence, refer to the module by model number and full serial number.

Mark the container FRAGILE to ensure careful handling.

If necessary to package the module in a container other than original packaging, observe the following (use of other packaging is not recommended):

- Wrap the module in heavy paper or anti-static plastic.
- Protect the front panel with cardboard.
- Use a double-wall carton made of at least 350-pound test material.
- Cushion the module to prevent damage.

Caution

Do not use styrene pellets in any shape as packing material for the module. The pellets do not adequately cushion the module and do not prevent the module from shifting in the carton. In addition, the pellets create static electricity which can damage electronic components.

2

Verifying the HP E2749

To verify the HP E2749

You can perform a quick verification of the basic functions of the HP E2749 using the built-in self-test function.

The test uses the command «ff–ff‟

See the chapter titled “SCPI Command Reference” for syntax and details.

If the test fails, the A1 assembly is probably faulty. See the chapter titled “Replacing Assemblies,” for information on replacing the module.

The “*TST?*” test does not test the module’s ability to transmit and receive over Fibre Channel. When the HP E2749 has been connected to the Fibre Channel loop or fabric, you can verify the ability to transmit and receive using the “DIAGNOSTIC:FIBRE:LOOPBACK” command. This command performs a loopback test. See the chapter titled “SCPI Command Reference” for more information.

3

Replacing Assemblies

Replaceable Parts

For information on upgrading your module or replacing parts, contact your local Hewlett-Packard sales and service office. See the inside back cover of this guide for a list of office locations and addresses.

Ordering Information

To order a part listed in the table, quote the Hewlett-Packard part number (HP Part Number) and the check digit (CD). Indicate the quantity required and address the order to the nearest Hewlett-Packard sales and service office (see the inside back cover of this guide). The check digit verifies that an order has been transmitted correctly, ensuring accurate and timely processing of the order. The first time a part is listed in the table, the quantity column (Qty) lists the total quantity of the part used in the module. For the corresponding name and location of the manufacturer's codes shown in the tables, see "Code Numbers."

Caution

The module is static sensitive. Use the appropriate precautions when removing, handling, and installing to avoid unnecessary damage.

Direct Mail Order System

Within the U.S.A., Hewlett-Packard can supply parts through a direct mail order system. Advantages of the Direct Mail Order System are:

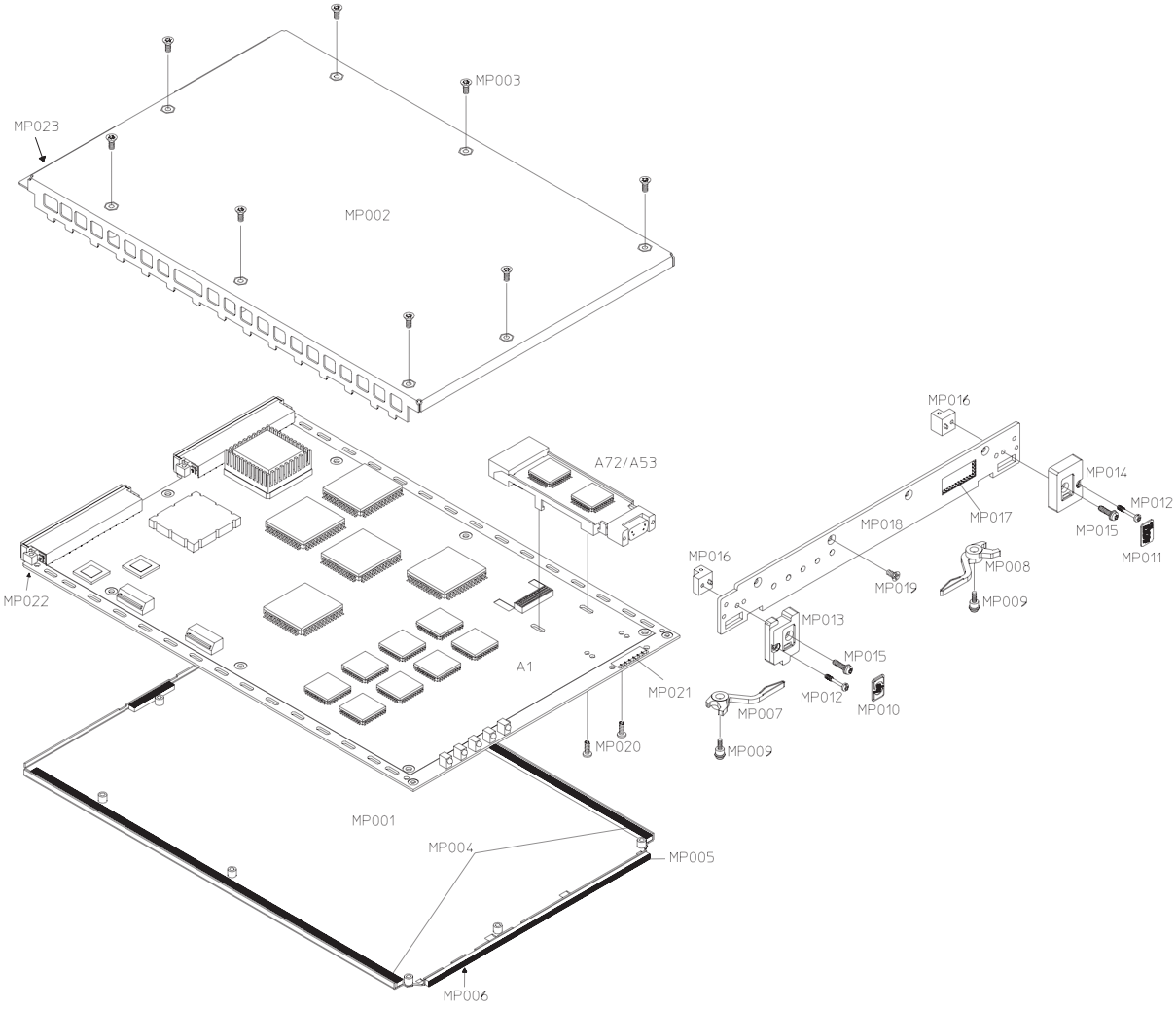
- Direct ordering and shipment from the HP Parts Center.
- No maximum or minimum on any mail order. There is a minimum order for parts ordered through a local HP sales and service office when the orders require billing and invoicing.
- Transportation charges are prepaid. A small handling charge is added to each order.
- No invoicing. A check or money order must accompany each order.
- Mail order forms and specific ordering information are available through your local Hewlett-Packard sales and service office. See the inside back cover of this guide for a list of Hewlett-Packard sales and service office locations and addresses.

Code Numbers

The following table provides the name and location for the manufacturers' code numbers (Mfr Code) listed in the replaceable parts tables.

Mfr No.	Mfr Name	Location
28480	Hewlett-Packard Company	Palo Alto, CA U.S.A.
30817	Instrument Specialties Co. Inc.	Placentia, CA U.S.A.
03647	Instrument Specialties Co. inc.	Del Water Gap, GA U.S.A.
01380	Amp Inc.	Harrisburg, PA U.S.A.

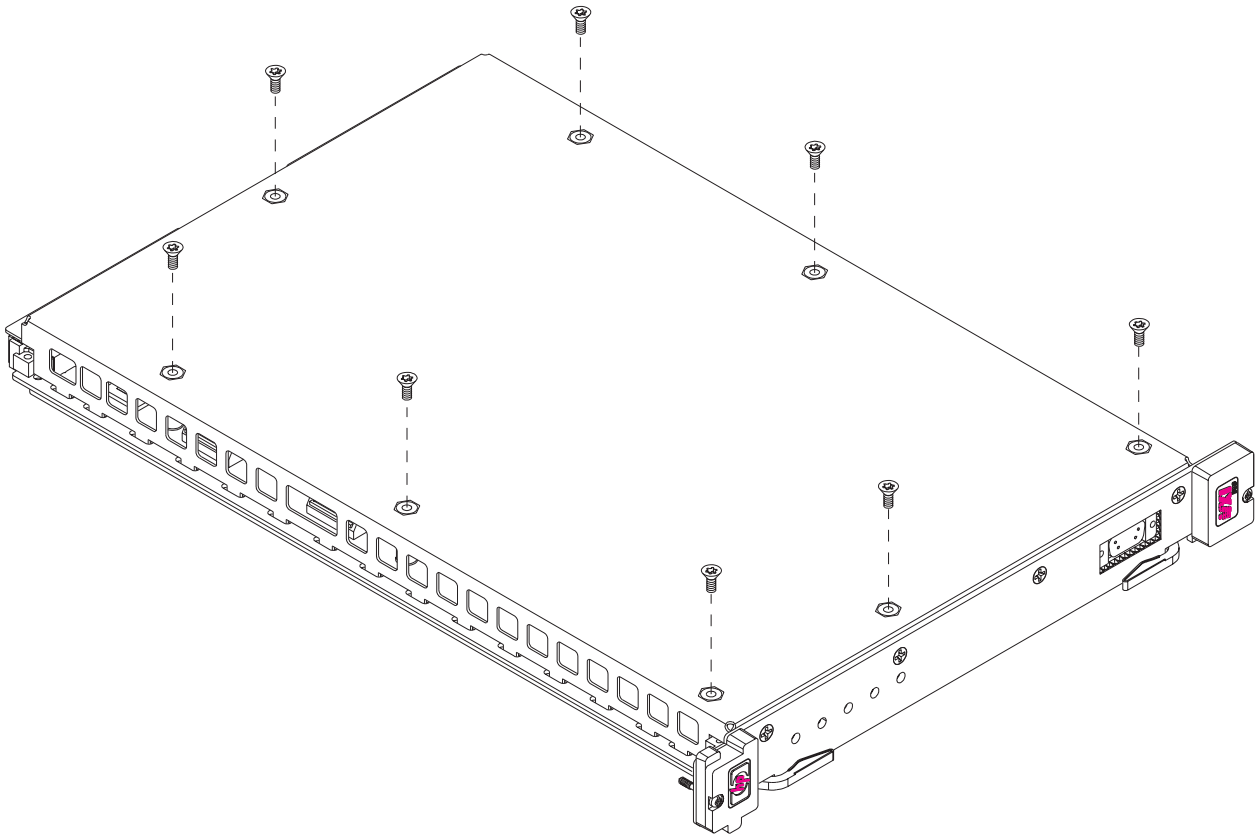
Assemblies


MP003
MP023
MP002
MP016
A72/A53
MP014
MP012
MP018
MP017
MP015
MP011
MP016
MP019
MP008
MP022
MP009
MP013
MP015
A1
MP021
MP012
MP010
MP007
MP020
MP009
MP001
MP004
MP005
MP006

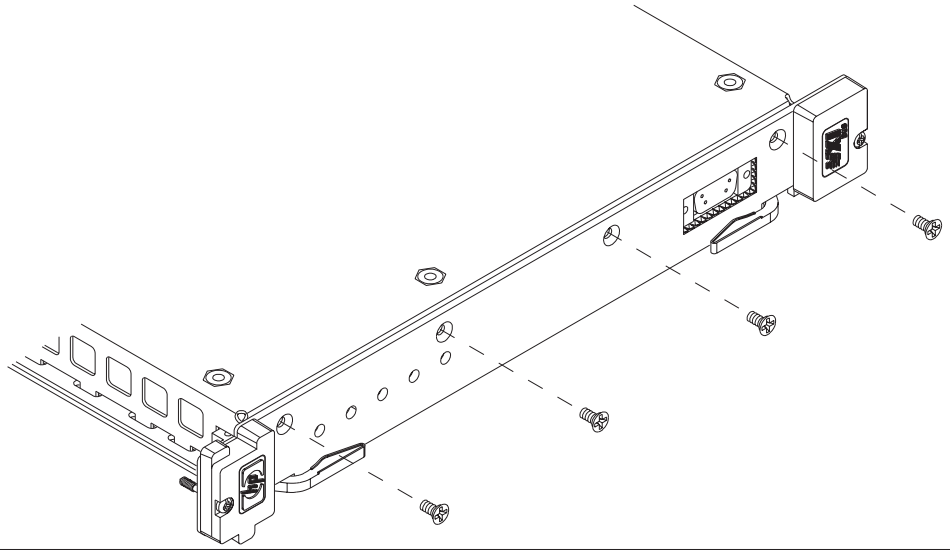
Ref Des	HP Part Number	CD	Qty	Description	Mfr Code	Mfr Part Number
A1	E2749-66501	5	1	PC ASSEM, EXCHANGE BRD.	28480	E2749-66501
A72	1005-0372	2	1	CON-MODULE COAX	28480	1005-0372
A53	1005-0353	9	1	CON-MODULE FIBER OP.	28480	1005-0353
MP001	E1437-00203	7	1	SHTF CVR-BTTM	28480	E1437-00203
MP002	E2749-00201	4	1	SHTF CVR-TOP	28480	E2749-00203
MP003	0515-1135	7	8	SCR-MCH M3.0 25M	28480	0515-1135
MP004	E1485-40601	1	2	GSKT-RFI, BOTTOM COVER	28480	E1485-40601
MP005	E1485-40602	2	2	GSKT RFI-FRONT PANEL	28480	E1485-40602
MP006	8160-0686	6	2	STMP FINGERS-RFI	30817	786-185
MP007	E1400-45101	5	1	MOLD, HANDLE LEFT	28480	E1400-45101
MP008	E1400-45102	6	1	MOLD, HANDLE RIGHT	28480	E1400-45102
MP009	E1400-00610	7	2	SCR-ASM SHLDR	28480	E1400-00610
MP010	E1400-84308	6	1	LABEL-HP, LOGO	28480	E1400-84308
MP011	E1400-84307	5	1	LABEL-VXI, LOGO	28480	E1400-84307
MP012	0515-2733	3	2	SCR-MCH M2.5 17	28480	0515-2733
MP013	E1400-45011	6	1	MOLD LBUT-ECL	28480	E1400-45011
MP014	E1400-45008	1	1	MOLD BOTTOM-LOGO	28480	E1400-45008
MP015	0515-0664	5	2	SCR-MCH M3.0 12M	28480	0515-0664
MP016	E1400-40104	8	2	CAST	28480	E1400-40104
MP017	8160-0683	3	4	STMP STRP-SPNG F	03647	0097-551-17-X
MP018	E2749-00203	6	1	PANEL-FRONT,"E2749A"	28480	E2749-00204
MP019	0515-1946	8	4	SCR-MCH M3.0 6MM	28480	0515-1946
MP020	0515-2508	0	2	SCR-TAP	28480	0515-2508
MP021	8160-0467	1	2	STMP RFI FNGRS 8	28480	8160-0467
MP022	E1450-01202	5	4	STMP SHLD-RFI GRND	28480	E1450-01202
MP023	8160-0634	4	2	STMP BE/CU GROUND	03647	0097-0611-17
	8120-8640	4	1	CBL-ASM DUPLX OPTION 010	01380	504971-2
	8120-8639	1	1	CBL-ASM CXL OPTION 011	01380	621771-3

To remove the top and bottom covers

1 Remove the eight screws from the top cover useing a T-10 torx driver.

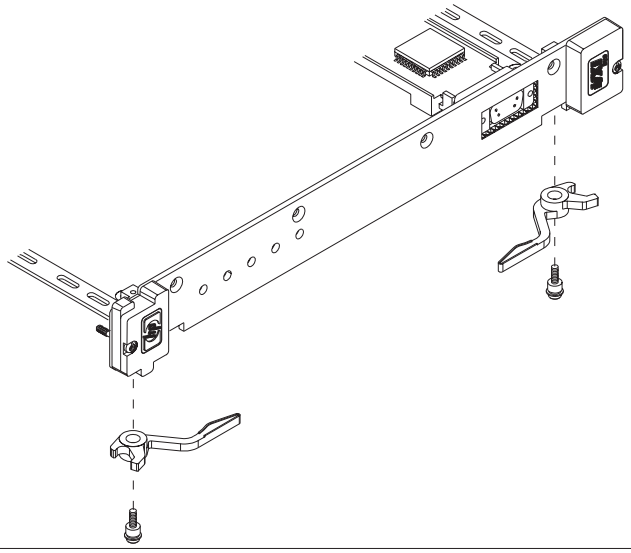


2 Remove the four screws that attach the top cover to the front panel using a T-10 torx driver. Remove covers.

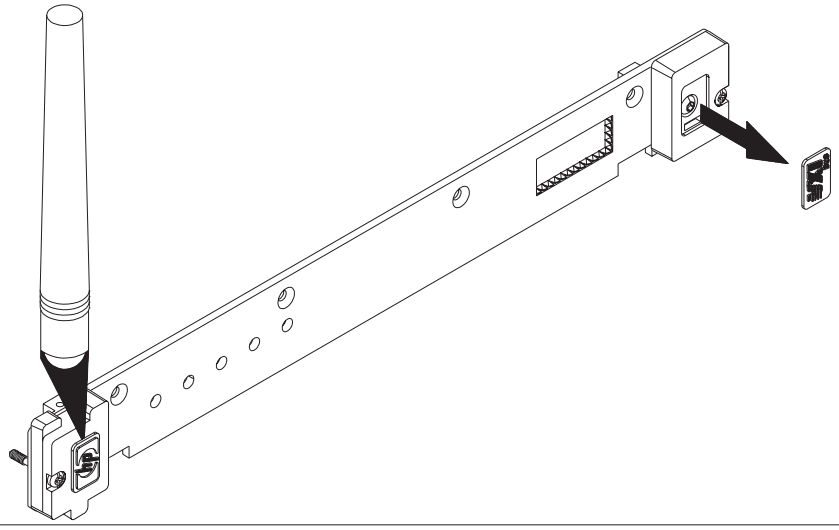


To remove the front panel

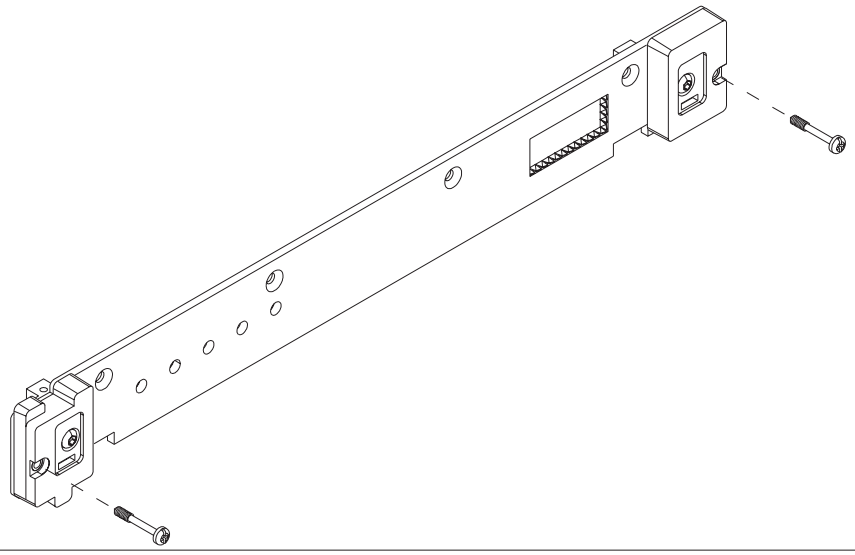
1 Remove covers (see “To remove the top and bottom covers”). Using a T-8 torx driver, remove the two screws that attach the handles to the assembly. *Note: be sure to label the two handles, which are different from each other. This will aid you in reassembling the module.*



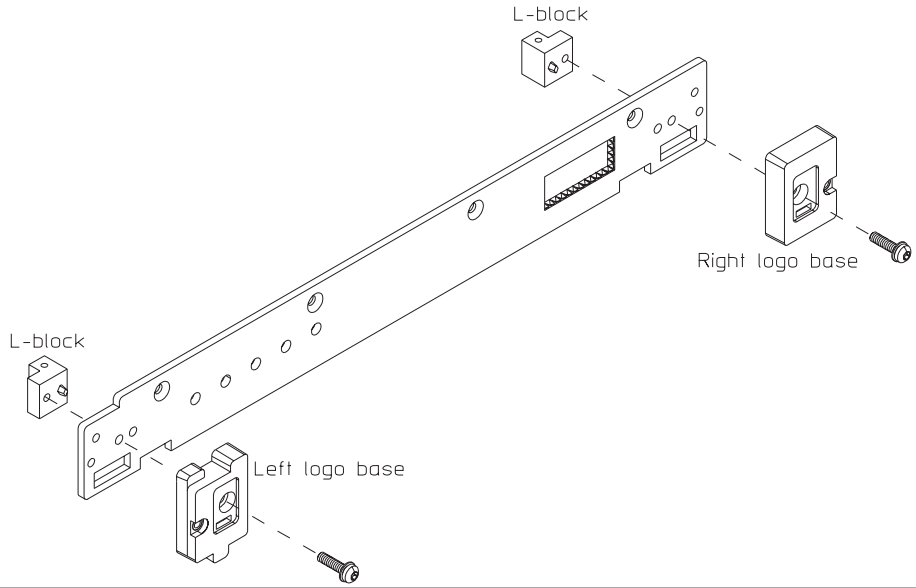
2 *Note: steps 2, 3, 4, and 5 are only necessary if you need to replace the front panel or any of its components.* Using an X-acto knife, gently pry the labels from the two keys



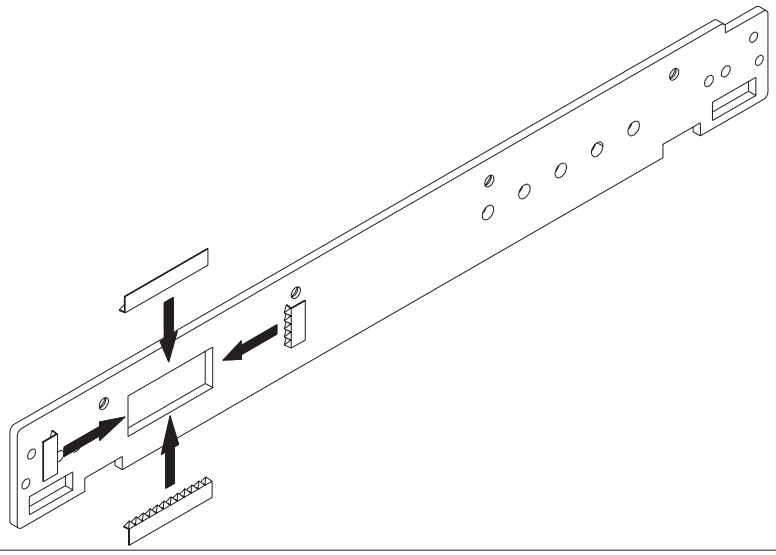
3 Using your hand, remove the two captive screws.



4 Using a T-10 torx driver, remove the two screws that attach the two logo bases and the two L-blocks to the front panel. *Note: there is a left and a right logo base. Also notice the orientation of the two L-blocks. This will be important when you reassemble the front panel.*

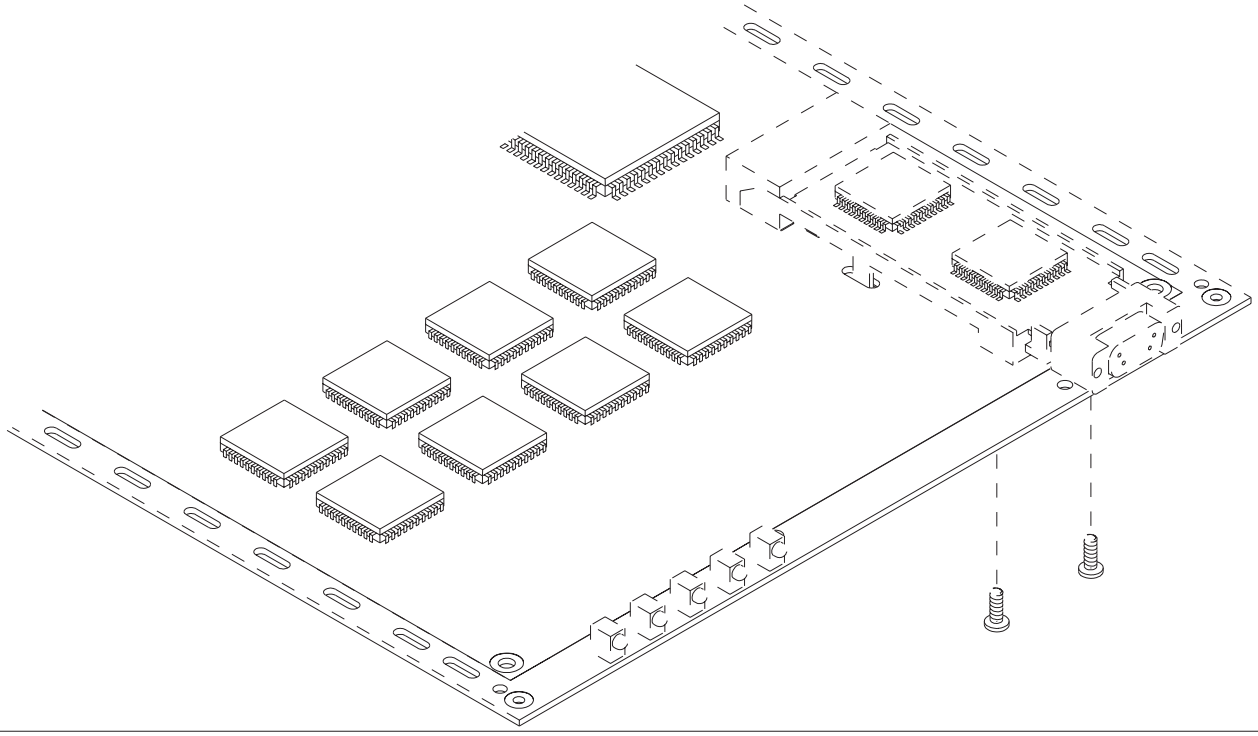


5 Using an X-acto knife, gently pry the RFI Strips from the rear of the front panel.
Note: there are12 sections on the top and bottom and 5 sections on each side. This may be important when ordering new strips.



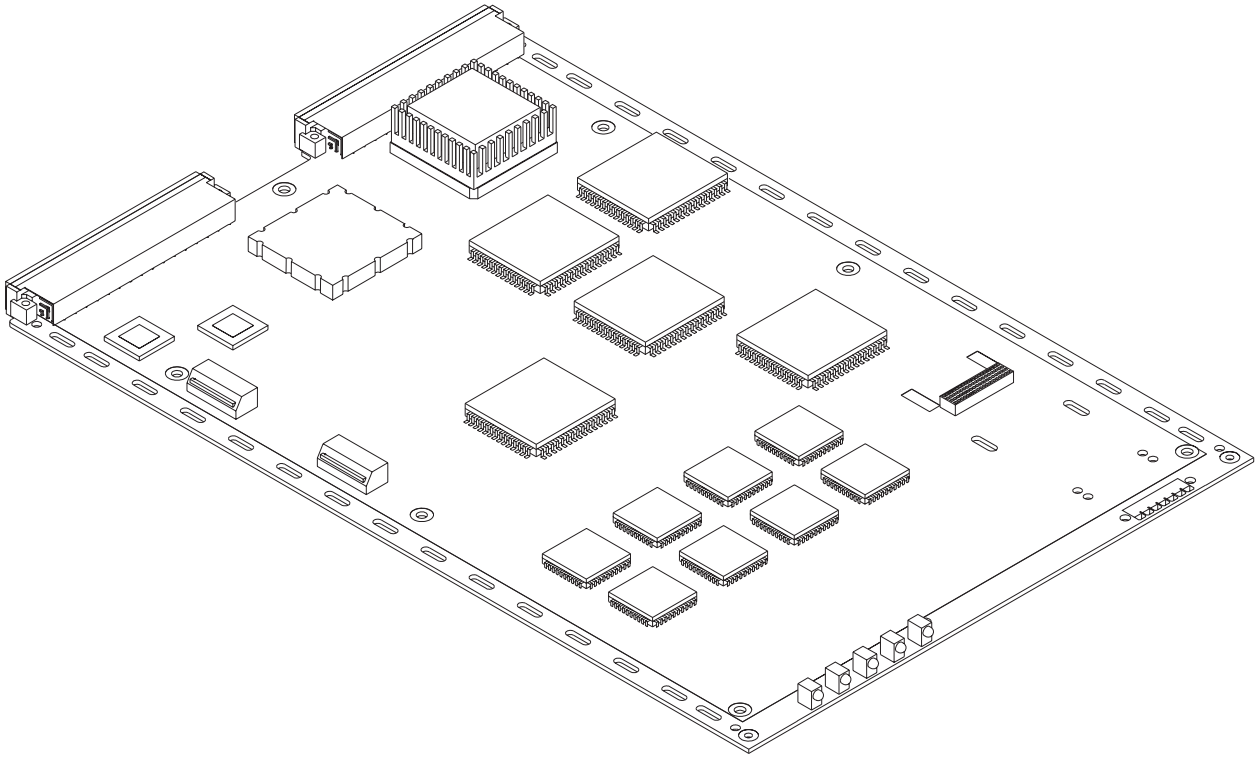
To remove the A72/A53 assembly (GLM)

1 Remove covers (see “To remove the top and bottom covers”). Remove the front panel (see step 1 of “To remove the front panel”). Using a T-8 torx driver remove the 2 screws that attach the A72/A53 (GLM) assembly to the A1 main assembly. Remove assembly. *Note: the illustration shows the A72 assembly, but the procedure is the same for both.*



To remove the A1 main assembly

1 Remove covers (see “To remove the top and bottom covers”). Remove the front panel (see step 1 of “To remove the front panel”). Remove the A72/A53 assembly (see “To remove the A72/A53 assembly).



4

Backdating

Backdating

This chapter will document modules that differ from those currently being produced. With the information provided in this chapter, this guide can be modified so that it applies to any earlier version or configuration of the module.

5

Hardware Description

General description

The HP E2749 Fibre Channel Interface Module occupies a single-wide C-size slot in a VXI mainframe. The module is available with either coaxial copper (standard) or optical (option 001) front panel connectors.

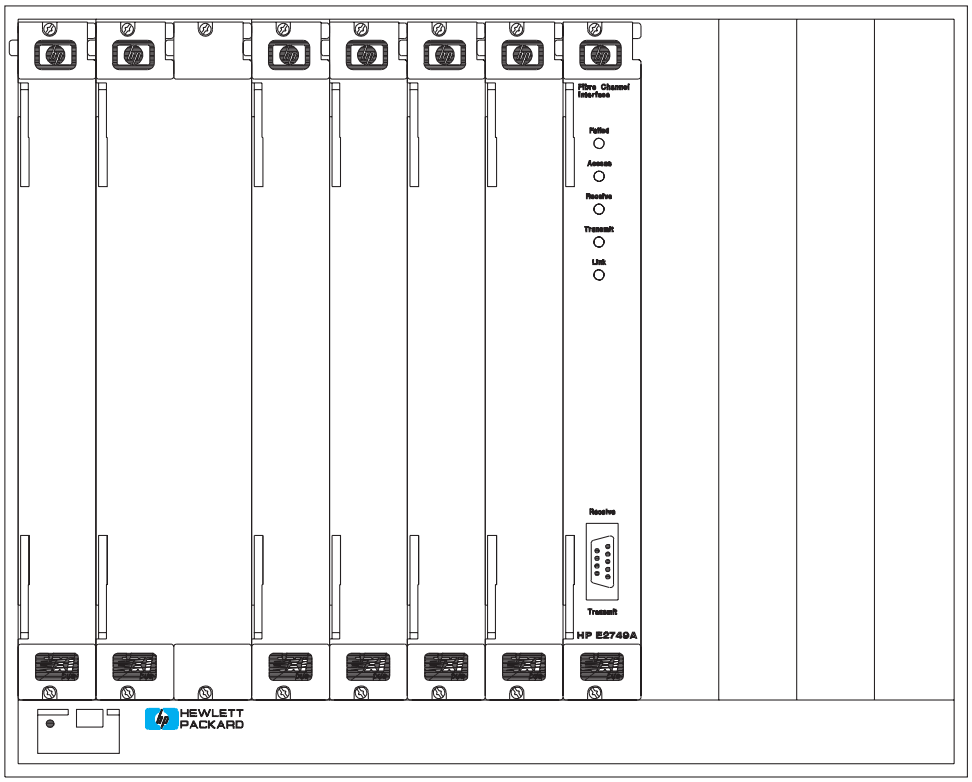


Figure 5-1: HP E2749 Fibre Channel Interface Module

Circuit description

The HP E2749 includes an HP Tachyon Fibre Channel Controller chip, an interface between the Tachyon chip and the local (P2) and VXI (P1) buses, and a CPU to direct the data transfer operations. The module receives and transmits data through a Gigabit Link Module (GLM) with either coaxial copper (standard) or optical (option 001) connectors. The GLM provides the Fibre Channel FC-0 layer solution (see "Fibre Channel Overview" for more information).

Data to be transferred over Fibre Channel can come from either the local bus, the VXI bus, via VME shared memory, or any combination of these sources, under control of a programmable sequencer built into the HP E2749. Dedicated hardware in the HP E2749 moves this data into a high-speed, dual port memory. From dual-port memory, the data goes to the HP Tachyon Fibre Channel chip, where data is encoded from 8-bit internal bytes into 10-bit Fibre Channel transmission characters. The encoded data passes through the GLM to Fibre Channel.

Data coming into the HP E2749 from Fibre Channel enters through the GLM to the HP Tachyon chip, where the data is decoded from 10-bit Fibre Channel characters into 8-bit internal bytes. From there the data goes to dual-port memory, then to the local bus, the VXI bus, or VME shared memory.

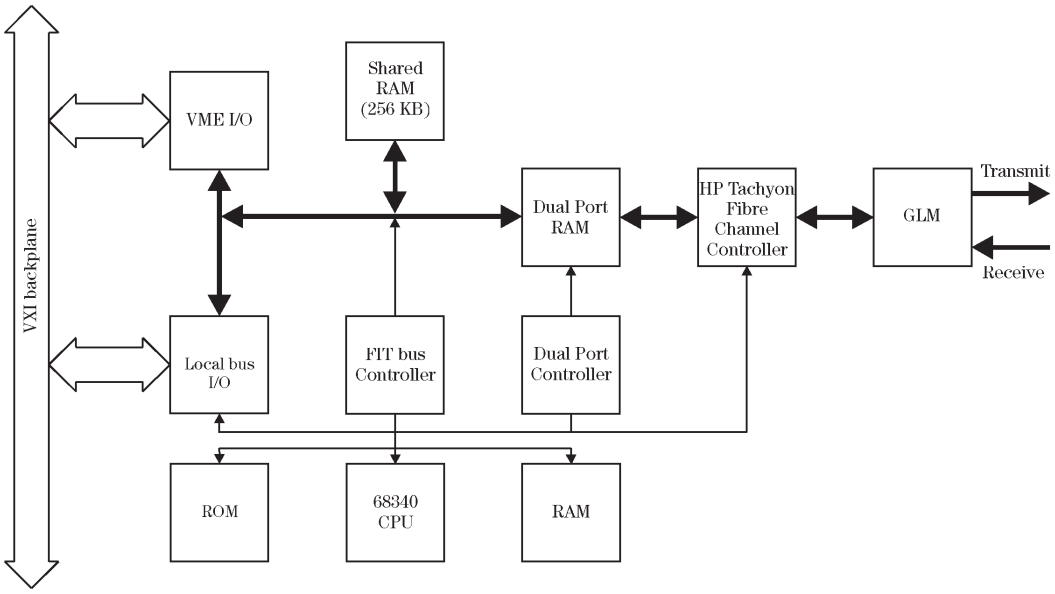


Figure 5-2: Block diagram of the HP E2749

HP E2749 front-panel description

Status LEDs

- *Failed* lights briefly when powering up and normally goes out after a few seconds. If it stays on it indicates a hardware failure in the module.
- Access lights when the module is being accessed via the VXI backplane.
- *Receive* lights when the module is receiving data via Fibre Channel.
- *Transmit* lights when the module is transmitting data via Fibre Channel.
- *Link* lights when a connection has been established between the module and another Fibre Channel device.

Receive/Transmit Connectors

The Receive/Transmit connectors on the HP E2749 are part of the gigabit per second link module (GLM) which interfaces between the Fibre Channel network and the main board.

The standard GLM has a coaxial copper interface with a 9-pin D-type connector.

The option 001 GLM has an optical interface with a duplex SC connector.

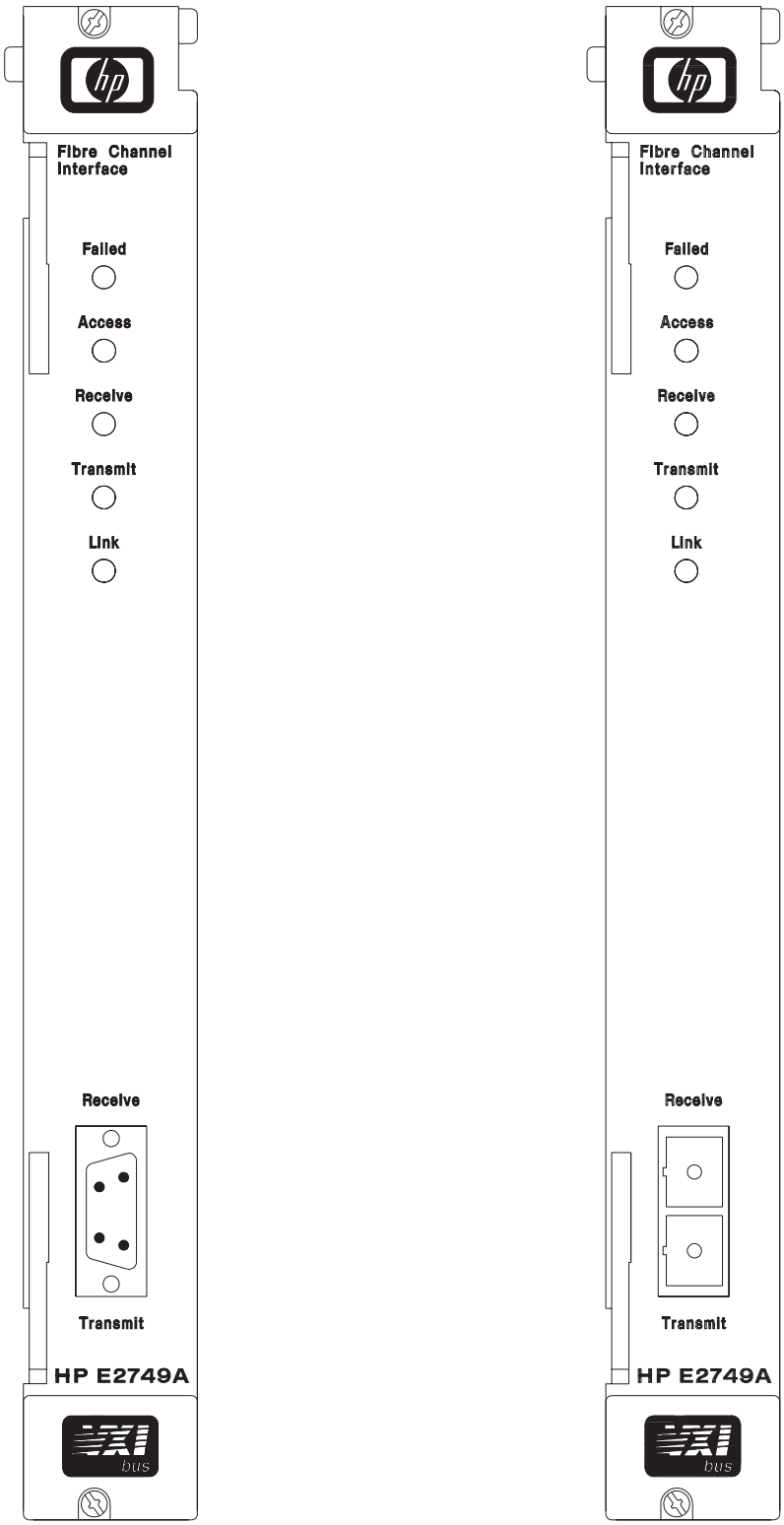


Figure 5-3: Front panel - HP E2749A standard and HP E2749A Option 001

6

Using the HP E2749

Introduction

One of the key features of VXI systems is the high-speed transfer of data (up to 100 Mbytes/s) over the local bus between modules in the same mainframe. Until recently, transferring data between VXI mainframes and other devices was considerably slower. A typical data transfer rate for a module on the VXIbus or MXI bus is 2 to 4 Mbytes/s. HP-IB has a maximum data transfer rate of about 1 Mbyte/sec, and the transfer rate might be even slower because HP-IB protocol limits the transfer speed to that of the slowest device on the bus.

The HP E2749 Fibre Channel Interface module provides rapid transfer of broadband data to and from a VXI mainframe. Now you can perform external data transfers at rates up to 41 Mbytes/sec. For example, the HP E2749A can transfer full-speed 16-bit data from an HP E1437A sampling at 20 Msamples/sec. This is equivalent to a transfer rate of 40 Mbytes/sec.

Some typical applications are transferring data between VXI mainframes, from a VXI mainframe to VME, or from a VXI mainframe to a PC.

Note

If you will be transferring data between VXI mainframes, each mainframe must include a controller in slot 0.

This chapter gives you a brief introduction to SCPI, VXI, and Fibre Channel. If you would like more information, here are a few publications you might find useful:

- *Feeling Comfortable with VXI,* available from Hewlett-Packard, publication number 5965-6497E.
- "An Introduction to Fibre Channel," Meryem Primmer, *HP Journal,* October 1996, 94-98.)
- *Fibre Channel: A Technical Description,* Fibre Channel Association, available from Systran Corporation, 4126 Linden Avenue, Dayton, OH 45432.
- Fibre Channel Association's Web site, http://www.amdahl.com/ext/CARP/FCA/FCA.html
- "Fibre Channel: Gigabit Communications and I/O for Computer Networks," Alan F. Benner, McGraw-Hill, ISBN 0-07-005669-2.
- *Feeling Comfortable with VXI,* available from Hewlett-Packard, publication number 5965-6497E.

HP E2749 and SCPI

You control the HP E2749 by sending commands using a language called *Standard Commands for Programmable Instruments*, or *SCPI*. SCPI is a closely defined, but broadly accepted, standard instrument command language. See chapter 8, “Programming the HP E2749 with SCPI,” and chapter 9, “SCPI Command Reference,” for detailed information on how to use SCPI to control the HP E2749.

SCPI commands

The following is an overview of the some of the capabilities of the HP E2749 controlled by SCPI commands. See chapter 8, “Programming the HP E2749 with SCPI” and chapter 9, “SCPI Command Reference” for details.

Sequence

The SEQuence subsystem, SEQuence[1|2|3|4], specifies the order of operations for transmitting or receiving data. Synchronization and control operations are provided for both transmit and receive.

The fields contained in every element of the Sequence list are: operation, count, address, and miscellaneous. The operation field specifies the action to be done: data transfer, synchronization, or control. The count field is used by many operations to indicate the size of the transfer. The unit of count is sometimes bytes and sometimes blocks — see the description of the operation to determine which. The address field is used by operations which do VME data transfers. The value of address is an offset from the beginning of one of the address spaces. The miscellaneous field has various meanings depending upon the operation. Not every operation uses all fields, but every Sequence element contains all four fields. Fields which are not used should be set to zero.

The Sequence operations labeled as synchronization or control may be used in either receive or transmit Sequences. They are intended to help provide synchronization between the Sequence and the devices or application program which generate/receive the data.

See the chapter titled “Using Sequence Operations with the HP E2749" for details on using Sequence commands.

Operation status register

The subsystem which refers to the operation status register is STATus:OPERation. These commands provide the necessary commands to interface with the operation register.

For more information about the operation status register and other status registers, see the chapter titled "Programming the HP E1562 with SCPI."

Address space

The VXI system architecture defines three types of address space. A16 space consists of 64 KBytes, A24 is 16 MBytes, and A32 is 4 GBytes.

The HP E2749 has access to A16, A24, and A32 space through a 16-bit port. Or, if devices support it, it can also use a 32-bit port using D32.

Shared memory

Shared memory provides a way for the HP E2749 to transfer data to a controller. The shared memory in the HP E2749 is mapped to the A24 VXI address space. The controller can then access that same address space to receive or write data. Note that if SCPI commands or Sequences refer to shared memory in the HP E2749, the addresses begin at zero. However if they refer to shared memory in the A24 space, they may begin at a different value, depending on how the A24 memory has been allocated among devices.

The VXI registers

The HP E2749 uses the following VXI registers:

- Offset Register
- Status/Control Register
- Device Type Register
- ID/Logical Address Register
- Data Low Register
- Response/Data Extended Register
- Protocol/Signal Register

These registers are common to all message-based VXI devices. Refer to Chapter 8, "Programming the HP E2749 with SCPI," or to VXI documentation for more information.

Fibre Channel overview

Fibre Channel is a standard, efficient, generic transport mechanism whose primary task is to transport data at the fastest possible speeds and with the least possible delay. It provides high-speed interconnection, communication, and data transfer among heterogeneous systems and peripherals, including workstations, mainframes, supercomputers, desktop computers, and storage devices. Fibre Channel handles both networking and peripheral I/O communication over a single channel using the same drivers, ports, and adapters for both types of communication.

Fibre Channel is structured as a set of hierarchical functions that support a number of existing protocols, but it does not have a native I/O command set. It is not a high-level protocol, but does contain a low-level protocol for managing link operations. Fibre Channel is not concerned with the content of user data being transported. Networking and I/O protocols are mapped to Fibre Channel constructs, then encapsulated and transported within Fibre Channel frames. The main purpose of Fibre Channel is to have any number of existing protocols operate over a variety of physical media and existing cable plants.

Fibre Channel has many advantages over existing communication systems:

- High speed—from 2.5 to 250 times faster than existing communications interfaces.
- Small connectors.
- Longer-distance operation—up to 500 m for the HP E2749.
- Multi-user—an addressing limit of 2^{24}, more than 16 million addresses.
- Protocol independent.

The next few pages contain an overview of Fibre Channel. For more detailed information, see the Fibre Channel publications listed in the introduction to this chapter.

Fibre Channel topologies

Fibre Channel supports three types of interconnection topologies: point-to-point, arbitrated loop, and fabric.

Point-to-point is a direct connection between two ports, such as between a VXI mainframe and a PC controller. This is the default interconnection.

Suppose you want to transfer data from an HP E1437 to a PC for post-processing. You would connect the VXI mainframe directly to a PC, a point-to-point connection.

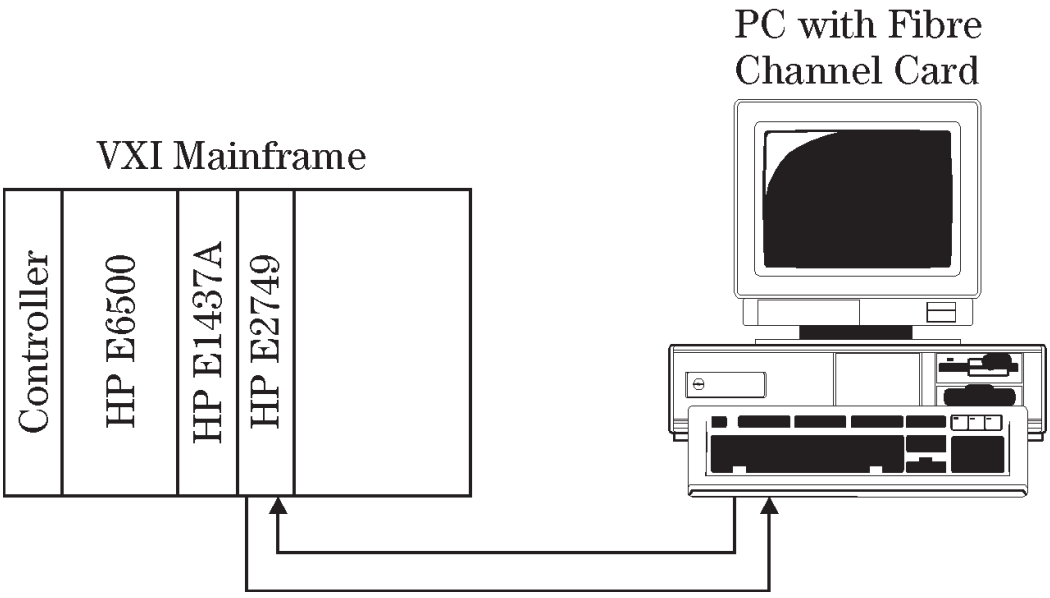


Figure 6-1: Fibre Channel point-to-point connection

An arbitrated loop connects multiple (up to 127) devices without hubs or switches. Each device makes a request for use of the loop when it needs to communicate with another device on the loop. When the loop is free, a bidirectional connection is established between the two devices. This is the most common type of Fibre Channel interconnection.

A Fibre Channel arbitrated loop might include a personal computer, a VME mainframe, and two VXI mainframes.

Data is transmitted from the source Fibre Channel module to Fibre Channel and received by the destination Fibre Channel module. Suppose you want to transfer data from the HP E1437 module in the lower VXI mainframe to the VME DSP module for post-processing. The HP E1437 data would pass through the HP E2749 module in the VXI mainframe, onto the arbitrated loop, and through the VME Fibre Channel module to the DSP module.

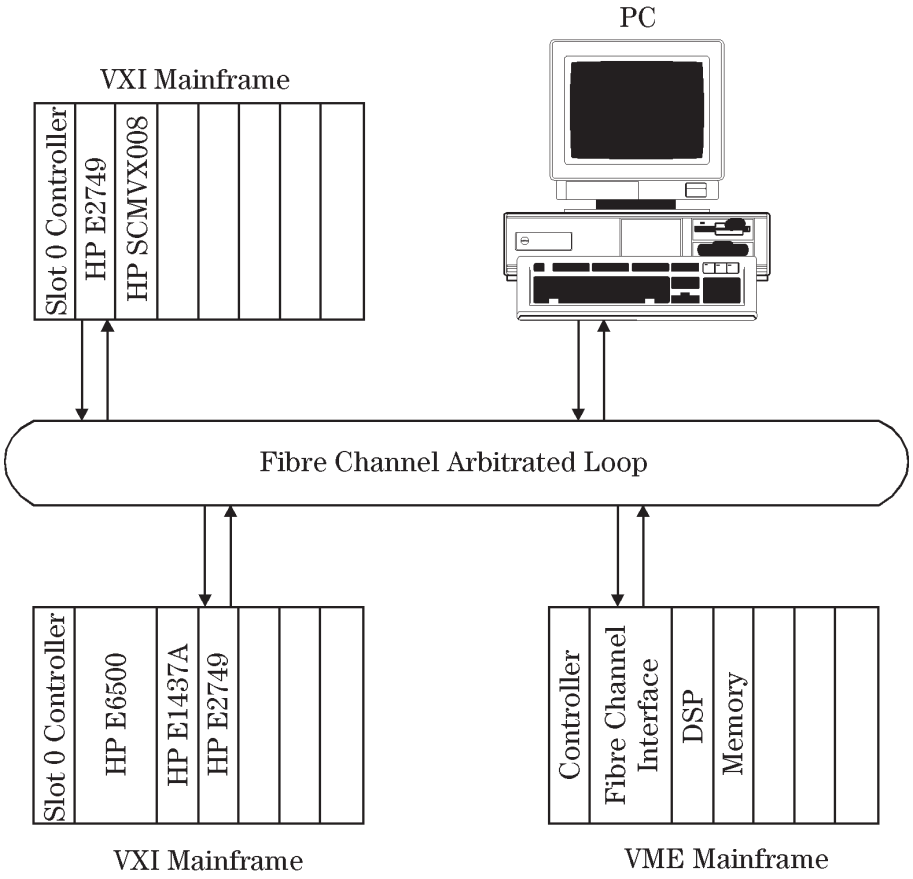


Figure 6-2: Fibre Channel arbitrated loop

The third topology is called a *fabric*. The fabric is one or more switching elements that operate much like a telephone exchange. Each device on the system connects to the fabric at a node, and the fabric manages the connections between nodes. A fabric theoretically can have up to 2^{24} nodes, so the fabric is the best topology for connecting large numbers of devices or for complex configurations.

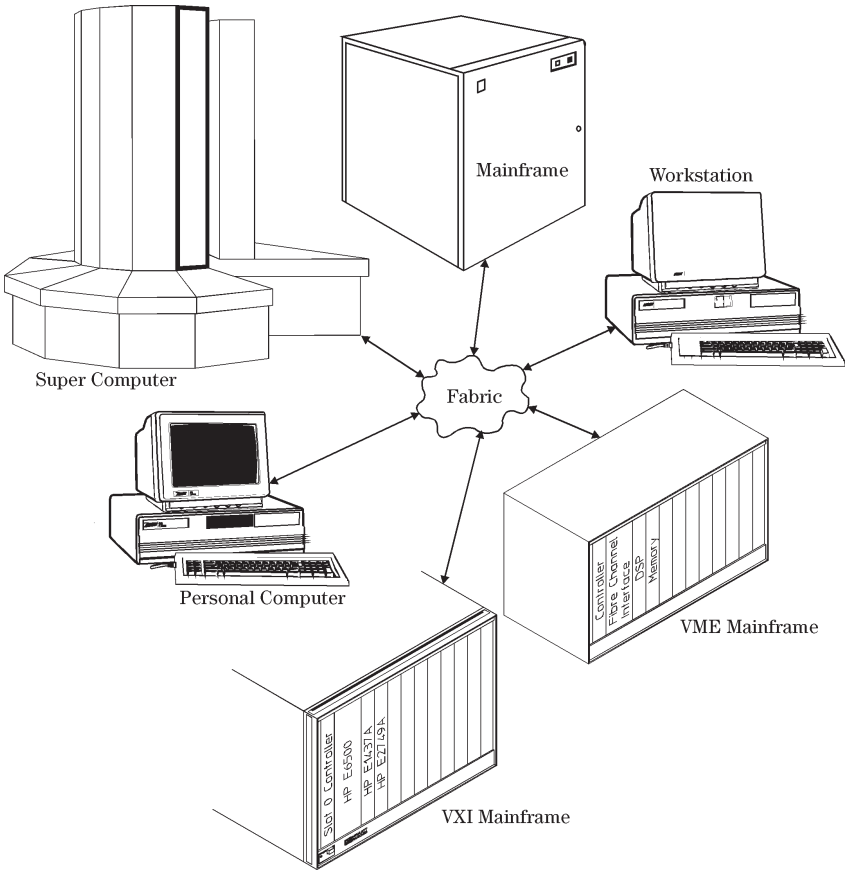


Figure 6-3: Fibre Channel fabric

Fibre Channel's multilevel structure

Fibre Channel is organized into five functional levels:

- FC-0 defines physical characteristics of the media for various signaling rates.
- FC-1 defines the transmission protocol, including data encoding and decoding, byte synchronization, and character-level error control.
- FC-2 defines the signaling protocol, including the hierarchical structure of frames, sequences, and exchanges, which is the data transport mechanism used by Fibre Channel. This level also manages the three classes of service provided by Fibre Channel.
- FC-3 defines a set of common services. This layer is still being formulated.
- FC-4 defines the mapping of upper level protocol (ULP) to services of the lower three layers. For example, the Fibre Channel protocol for SCSI, known as FCP, defines a Fibre Channel mapping layer that uses the services of the lowest three Fibre Channel layers to transmit SCSI command, data, and status information between a SCSI initiator and a SCSI target.

The lowest three levels define the Fibre Channel Physical standard (FC-PH). FC-3 and FC-4 handle interfaces between FC-PH and other network protocols and applications. The HP E2749 provides all of FC-PH and layer FC-4.

The HP E2749A supports only one upper level protocol—Systran Corporation's FibreXpress Lightweight Protocol (FXLP).

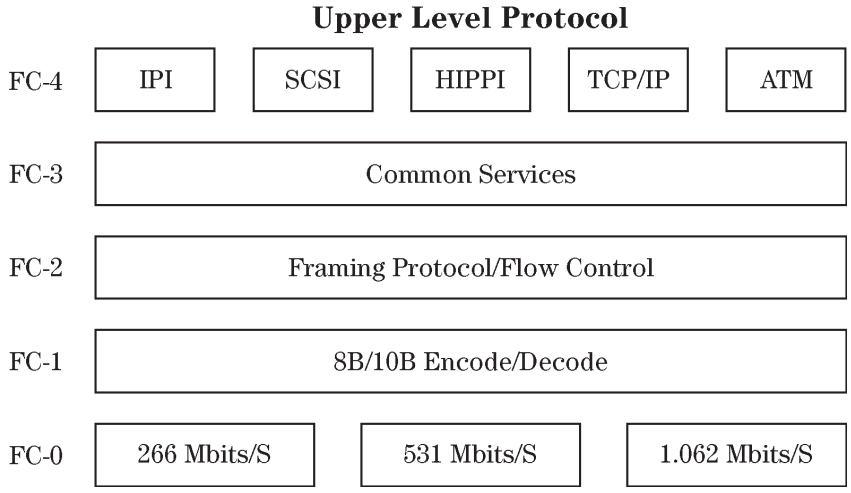


Figure 6-4: Fibre Channel's 5 layers

7

Using Sequence Operations with the HP E2749

Sequence overview

What is a Sequence?

A Sequence is a list of data transfer, control, and synchronization operations. Each data transfer operation identifies the source or destination of data in the VXI chassis and the amount of data to be transferred. Control and synchronization operations are used to coordinate the activity of Sequence with the application program or with hardware modules installed in the VXI chassis.

Transmit Sequences are used to collect data from modules in the VXI chassis using the VXI bus or the Local Bus and transmit the data over Fibre Channel to a receiving application.

Receive Sequences are used to distribute data received from a transmitting application over Fibre Channel to modules in the VXI chassis using the VXI bus or the Local Bus. A receive Sequence can be used to either distribute received data over the Local Bus or the VXI bus but not both.

A Sequence may perform either transmit or receive operations; transmit and receive operations cannot be mixed within the same sequence. The behavior of a receive operation encountered when using the sequence to transmit is undefined. Conversely, the behavior of a transmit operation encountered when using the sequence to receive is undefined. The behavior of a receive sequence is also undefined if it contains both VXI bus and Local Bus operations. Both transmit and receive Sequences may contain control, and synchronization operations.

Loading and starting Sequences

Up to four Sequences may be loaded into the HP E2749 but only one can run at a time. A Sequence is loaded into the HP E2749 using the "SEQuence:ADD" SCPI command. The Sequence is started using the "SEQuence:BEGin" SCPI command. The first parameter of the "SEQ:BEG" command determines whether the sequence is being used to transmit or receive data and the second parameter identifies the total number of bytes to transfer. If the total number of bytes to transfer specified in the "SEQ:BEG" command is greater than the total number of bytes specified in the list of data transfer operations in the sequence, the sequence will be executed multiple times until the transfer count from "SEQ:BEG" command is satisfied.

Creating Sequences

Adding Sequence elements

Sequence operations are loaded into the HP E2749 by using the SCPI command:

→□ᵂ₊⅝–⅛⅝Frᴼ◆½◆¼◆¢ᴳ¾“‹‹ ’■ᴴ₊⅝ᶜᵣ⅓ᴺ∟ⓘ¹–‘£’—¹ᵛ₊–ᴺ∟‘£’“⅜⅜ᶜᵣ⅝ᴸꜰᴸꜰ‘£’●ⓘᴸꜰ⅛‘

Each time this command is issued an element representing one operation is added to the end of the Sequence queue in memory. The maximum number of operations in a single sequence is 100.

If the Sequence number [1|2|3|4] is not specified the Sequence element is added to Sequence 1.

Required fields

Every Sequence element requires that all four fields: (<Operation>,<Count>,<Address>,<Misc>) be filled though not every operation uses all fields. For some Sequence operations certain fields represent two pieces of information as indicated in the Sequence operation descriptions.

The `<Operation>` field specifies what type of action will take place: data transfer, synchronization, or control. This value corresponds to the code listed in the programming reference section of this chapter for the specific type of operation.

The `<Count>` field is used by many operations to indicate how many units will be transferred. The unit of <Count> may be either bytes or blocks, as indicated in the description of each operation. For some Sequence operations this field represents two pieces of information as indicated in the Sequence operation descriptions.

The `<Address>` field is used mainly by operations which transfer data over the VXI System Bus. The value of `<Address>` is an offset from the beginning of one of the address spaces. The Shared RAM space is local to the HP E2749.

The miscellaneous `<Misc>` field has various meanings depending on the operation.

Accepted Field Values

Field values must be specified as numeric values. All decimal representations, including signs, decimal points, and scientific notation are accepted as field values:

123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01

Note, however, that negative numbers will generate an error and fractional values will be automatically rounded to the nearest integer.

The fields may be specified in decimal, hex (#h prefix), octal (#q prefix), or binary (#b prefix):

123, #h7B, #q173, #b1111011

Related SCPI commands

Other SCPI commands in addition to SEQuence:ADD which can or must be used with relation to Sequence operations are documented in detail in the SCPI programming section of this book. These commands include:

- MMEMory:SESSion:PROTocol which selects the communication protocol to be used for the data transfer.
- SEQuence:BEGin which starts Sequence execution.
- SEQuence:DELete:ALL which deletes all operations from the current Sequence list. This command should be sent before adding elements to a Sequence.
- SEQuence:SIZE? which returns the number of elements in the Sequence.

Sequence quick reference

	Operation	Code (hex)	Count	Address	Misc
Control operations					
	Do nothing	0000	N/A	N/A	N/A
	Terminate Sequence	0001	N/A	N/A	N/A
	Pause N msec	0002	Milliseconds	N/A	N/A
	Pause N loops	000a	Loops	N/A	N/A
	Execute new Sequence	0004	Seq nbr	N/A	N/A
	New Sequence if count	0005	Seq nbr	MSB	LSB
	TTLTRG control	0003	Bit field	N/A	N/A
	TTLTRG arm	0006	Bit field	N/A	N/A
	TTLTRG wait	0007	Bit field	N/A	N/A
	IRQ arm	0008	Bit field	N/A	N/A
	IRQ wait	0009	Bit field	N/A	N/A
	Test shared RAM Skip	7000	N/A	RAM address	Skip count
Synchronization operations					
	Wait Bit Set A16	6000	Bit mask	A16 address	Loops
	Wait Bit Clear A16	6001	Bit mask	A16 address	Loops
	Wait Bit Set A24	6002	Bit mask	A24 address	Loops
	Wait Bit Clear A24	6003	Bit mask	A24 address	Loops
	Wait Bit Set A32	6004	Bit mask	A32 address	Loops
	Wait Bit Clear A32	6005	Bit mask	A32 address	Loops
	Wait Bit Set Shared RAM	6006	Bit mask	RAM address	Loops
	Wait Bit Clear Shared RAM	6007	Bit mask	RAM address	Loops
	Wait A16 Count 16	6008	16 bit value	A16 address	Loops
	Wait A24 Count 16	6009	16 bit value	A24 address	Loops
	Wait A32 Count 16	600A	16 bit value	A32 address	Loops
	Wait Count Shared RAM 16	600B	16 bit value	RAM address	Loops

	Operation	Code (hex)	Count	Address	Misc
	Wait A16 Count 32	600C	32 bit value	A16 address	Loops
	Wait A24 Count 32	600D	32 bit value	A24 address	Loops
	Wait A32 Count 32	600E	32 bit value	A32 address	Loops
	Wait Count Shared RAM 32	600F	32 bit value	RAM address	Loops
	Control A16 Reg16	6018	N/A	A16 address	Value
	Control A24 Reg16	6019	N/A	A24 address	Value
	Control A32 Reg16	601A	N/A	A32 address	Value
	Control Reg Shared RAM 16	601B	N/A	RAM address	Value
	Control A16 Reg 32	601C	N/A	A16 address	Value
	Control A24 Reg 32	601D	N/A	A24 address	Value
	Control A32 Reg 32	601E	N/A	A32 address	Value
	Control Reg Shared RAM 32	601F	N/A	RAM address	Value
	Dump A24 Seq Bytes	6020	N/A	A24 address	N/A
	Dump A32 Seq Bytes	6021	N/A	A32 address	N/A
	Dump Shared RAM Seq Bytes	6022	N/A	RAM address	N/A
Transmit local bus data to Fibre Channel					
	Lbus Consume	1000	Lbus blocks	N/A	Lbus width-Bytes/block
	Lbus Eavesdrop	1001	Lbus blocks	N/A	Lbus width-Bytes/block
Transmit VXI bus data to Fibre Channel					
	Read A16 Buff 16	3000	Transfer bytes	A16 address	N/A
	Read A16 Buff 32	3001	Transfer bytes	A 16 address	N/A
	Read A16 Buff D32	3002	Transfer bytes	A16 address	N/A
	Read A24 Buff 16	3003	Transfer bytes	A24 address	N/A
	Read A24 Buff 32	3004	Transfer bytes	A24 address	N/A
	Read A24 Buff D32	3005	Transfer bytes	A24 address	N/A
	Read A32 Buff 16	3006	Transfer bytes	A32 address	N/A
	Read A32 Buff 32	3007	Transfer bytes	A32 address	N/A

Operation	Code (hex)	Count	Address	Misc
Read A32 Buff D32	3008	Transfer bytes	A32 address	N/A
Read A16 FIFO 16	3009	Transfer bytes	A16 address	N/A
Read A16 FIFO 32	300A	Transfer bytes	A16 address	N/A
Read A16 FIFO D32	300B	Transfer bytes	A16 address	N/A
Read A24 FIFO 16	300C	Transfer bytes	A24 address	N/A
Read A24 FIFO 32	300D	Transfer bytes	A24 address	N/A
Read A24 FIFO D32	300E	Transfer bytes	A24 address	N/A
Read A32 FIFO 16	300F	Transfer bytes	A32 address	N/A
Read A32 FIFO 32	3010	Transfer bytes	A32 address	N/A
Read A32 FIFO D32	3011	Transfer bytes	A32 address	N/A
Read Shared RAM	3012	Transfer bytes	RAM address	N/A
Dummy Bytes	3100	Pad bytes	N/A	N/A
Transmit local bus data to Fibre Channel, and monitor the data				
Lbus Consume Monitor Shared RAM	5000	Lbus blocks	RAM address	Lbus width-Bytes/block
Lbus Eavesdrop Monitor Shared RAM	5001	Lbus blocks	RAM address	Lbus width-Bytes/block
Lbus Consume Monitor A24	5014	Lbus blocks	A24 address	Lbus width-Bytes/block
Lbus Eavesdrop Monitor A24	5015	Lbus blocks	A24 address	Lbus width-Bytes/block
Transmit VXI bus data to Fibre Channel, and monitor the data				
Read Shared RAM Monitor Shared RAM	3812	Monitor bytes	RAM address	RAM address
Read Shared RAM Monitor A24 Buff D32	3912	Monitor bytes	RAM address	A24 address
Read Shared RAM Monitor A24 Buff	3a12	Monitor bytes	RAM address	A24 address
Read A16 FIFO D32 Monitor Shared RAM	380b	Monitor bytes	A16 address	RAM address
Read A16 FIFO D32 Monitor A24 Buff D32	390b	Monitor bytes	A16 address	A24 address
Read A16 FIFO D32 Monitor A24 Buff	3a0b	Monitor bytes	A16 address	A24 address
Read A16 FIFO16 Monitor Shared RAM	3809	Monitor bytes	A16 address	RAM address
Read A16 FIFO16 Monitor A24 Buff D32	3909	Monitor bytes	A16 address	A24 address

	Operation	Code (hex)	Count	Address	Misc
	Read A16 FIFO16 Monitor A24 Buff	3a09	Monitor bytes	A16 address	A24 address
	Read A16 Buff 16 Monitor Shared RAM	3800	Monitor bytes	A16 address	RAM address
	Read A16 Buff 16 Monitor A24 BuffD32	3900	Monitor bytes	A16 address	A24 address
	Read A16 Buff 16 Monitor A24 Buff	3a00	Monitor bytes	A16 address	A24 address
	Read A16 Buff D32 Monitor Shared RAM	3802	Monitor bytes	A16 address	RAM address
	Read A16 Buff D32 Monitor A24 Buff D32	3902	Monitor bytes	A16 address	A24 address
	Read A16 Buff D32 Monitor A24 Buff	3a02	Monitor bytes	A16 address	A24 address
	Read A24 FIFO D32 Monitor Shared RAM	380e	Monitor bytes	A24 address	RAM address
	Read A24 FIFO D32 Monitor A24 Buff D32	390e	Monitor bytes	A24 address	A24 address
	Read A24 FIFO D32 Monitor A24 Buff	3a0e	Monitor bytes	A24 address	A24 address
	Read A24 FIFO 16 Monitor Shared RAM	380c	Monitor bytes	A24 address	RAM address
	Read A24 FIFO 16 Monitor A24 Buff D32	390c	Monitor bytes	A24 address	A24 address
	Read A24 FIFO 16 Monitor A24 Buff	3a0c	Monitor bytes	A24 address	A24 address
	Read A24 Buff 16 Monitor Shared RAM	3803	Monitor bytes	A24 address	RAM address
	Read A24 Buff 16 Monitor A24 Buff D32	3903	Monitor bytes	A24 address	A24 address
	Read A24 Buff 16 Monitor A24 Buff	3a03	Monitor bytes	A24 address	A24 address
	Read A24 Buff D32 Monitor Shared RAM	3805	Monitor bytes	A24 address	RAM address
	Read A24 Buff D32 Monitor A24 Buff D32	3905	Monitor bytes	A24 address	A24 address
	Read A24 Buff D32 Monitor A24 Buff	3a05	Monitor bytes	A24 address	A24 address
Receive data from Fibre Channel and write to local bus					
	Lbus Generate	2000	Lbus blocks	N/A	Lbus width-Bytes/block
	Lbus Append	2001	Lbus blocks	N/A	Lbus width-Bytes/block
Receive data from Fibre Channel and write to VXI bus					
	Write A16 Buff 16	4000	Transfer bytes	A16 address	N/A
	Write A16 Buff 32	4001	Transfer bytes	A16 address	N/A
	Write A16 Buff D32	4002	Transfer bytes	A16 address	N/A
	Write A24 Buff 16	4003	Transfer bytes	A24 address	N/A

Operation	Code (hex)	Count	Address	Misc
Write A24 Buff 32	4004	Transfer bytes	A24 address	N/A
Write A24 Buff D32	4005	Transfer bytes	A24 address	N/A
Write A32 Buff 16	4006	Transfer bytes	A32 address	N/A
Write A32 Buff 32	4007	Transfer bytes	A32 address	N/A
Write A32 Buff D32	4008	Transfer bytes	A32 address	N/A
Write A16 FIFO 16	4009	Transfer bytes	A16 address	N/A
Write A16 FIFO 32	400A	Transfer bytes	A16 address	N/A
Write A16 FIFO D32	400B	Transfer bytes	A16 address	N/A
Write A24 FIFO 16	400C	Transfer bytes	A24 address	N/A
Write A24 FIFO 32	400D	Transfer bytes	A24 address	N/A
Write A24 FIFO D32	400E	Transfer bytes	A24 address	N/A
Write A32 FIFO 16	400F	Transfer bytes	A32 address	N/A
Write A32 FIFO 32	4010	Transfer bytes	A32 address	N/A
Write A32 FIFO D32	4011	Transfer bytes	A32 address	N/A
Write Shared RAM	4012	Transfer bytes	RAM address	N/A
Bit Bucket	4100	Discard bytes	N/A	N/A

HP E2749 Sequence operations

The following pages contain detailed descriptions of each sequence operation. The operations are listed in numerical (code) order. If you do not know the operation code, there are two ways to find the operation description:

- Scan through the quick reference tables beginning on page 7-6. The operations are listed by type, and the code is in the second column.
- Look up the operation name in the index, which will refer you to the correct page in this chapter.

Do Nothing 0000

No Sequence operation is performed

Sequence Syntax: #h0000,<Count>,<Address>,<Misc>

<Count> ::= 0

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0000,0,0,0

Description: No fields are used.

Terminate Sequence 0001

The Sequence stops executing.

Sequence Syntax: #h0001,<Count>,<Address>,<Misc>

<Count> ::= 0

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0001,0,0,0

Description: This operation terminates the Sequence even if the final count has not been met. This is useful only for creating a non-looping or one-time Sequence. No fields are used.

Pause N msec 0002

The Sequence stops executing for a designated period of time.

Sequence Syntax: #h0002,<Count>,<Address>,<Misc>

<Count> ::= 10:4294967295

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0002,40,0,0

Description: This operation causes the Sequence to stop executing for the number of milliseconds designated by <Count>. The resolution of the clock is only 10 msec, therefore the specified count is rounded to the nearest 10 msec value. For a pause of shorter duration see the 'Pause N loops' operation (000a).

<Address> and <Misc> are not used.

TTLTRG Control 0003

Controls the assertion of the TTLTRG lines.

Sequence Syntax: #h0003,<Count>,<Address>,<Misc>

<Count> ::= 0:#b11111111

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0003,#b1010101,0,0

Description: All TTLTRG lines are controlled simultaneously. Therefore, one (or more) lines may be set while all others are cleared. Any bits set to 1 in bits 0-7 of <Count> represent corresponding TTLTRG lines which are asserted.

<Address> and <Misc> are not used.

See the related TTLTRG Arm (0006) and TTLTRG Wait (0007) operations.

Execute New Sequence 0004

Begins executing a new logical Sequence.

Sequence Syntax: #h0004,<Count>,<Address>,<Misc>

<Count> ::= 1:4

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0004,3,0,0

Description: This operation begins executing the new logical Sequence specified by <Count>. The new Sequence inherits the Sequence type and total bytes remaining from the currently executing Sequence. This operation is useful in situations which require a one-time set of operations at the beginning of a transmit followed by a looping set of data acquisition operations. An example of such a one-time action is writing a header at the beginning of a data stream.

<Address> and <Misc> are not used.

New Sequence If Count 0005

Begins executing a new logical Sequence if the remaining byte count is less than the value specified.

Sequence Syntax: #h0005,<Count>,<Address>,<Misc>

<Count> ::= 1:4

<Address> ::= 0:#hFFFFFFFF

<Misc> ::= 0:#hFFFFFFFF

SCPI example: SEQ:ADD #h0005,2,#hAEC,#h33E1F671

Description: This operation begins executing the new logical Sequence number specified by <Count> if the remaining byte count is less than that specified by <Address> and <Misc>. Since the byte count is a 64-bit value and the Sequence fields are only 32-bit values, both the <Address> and <Misc> are used to specify the byte count. The most significant 32 bits are specified in the <Address> field and the least significant 32 bits are specified in the <Misc> field. The new Sequence inherits the Sequence type and total bytes remaining from the currently executing Sequence.

TTLTRG Arm 0006

Clears a set of latched TTLTRG assertions.

Sequence Syntax: #h0006,<Count>,<Address>,<Misc>

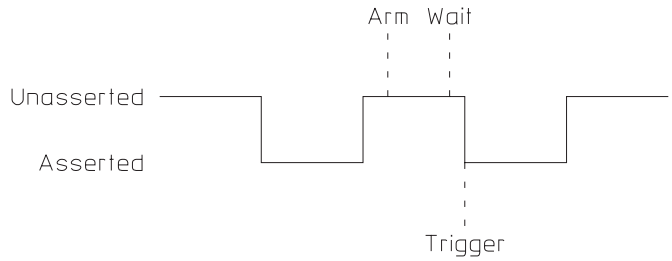
<Count> ::= 0:#b11111111

<Address> ::= 0

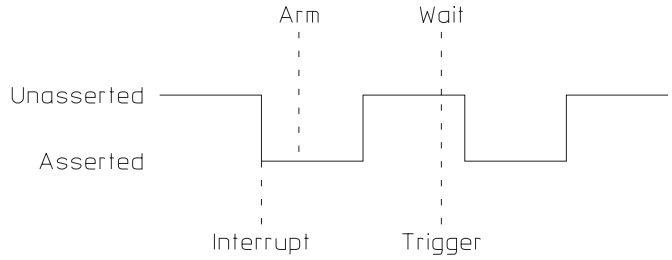
<Misc> ::= 0

SCPI example: SEQ:ADD #h0006,#b11011101,0,0

Description: Clearing latched TTLTRG assertions guarantees that any subsequent TTLTRG Wait will not be satisfied by an old latched TTLTRG assertion. Any bits set to 1 in bits 0-7 of <Count> clear assertions for corresponding to TTLTRG lines. The diagram below illustrates the effect of TTLTRG Arm and TTLTRG Wait on triggering in response to TTLTRG line assertion:



A perceived exception occurs if the trigger line is already asserted (set to the low voltage level) when the TTLTRG arm command is issued. In this case a subsequent TTLTRG Wait will result in no delay because the assertion requirement was previously fulfilled by the interrupt generated prior to TTLTRG Arm:



<Address> and <Misc> are not used.

See the related TTLTRG Control (0003) and TTLTRG Wait (0007) operations.

TTLTRG Wait 0007

Waits for a set of TTLTRG lines to be asserted.

Sequence Syntax: #h0007,<Count>,<Address>,<Misc>

<Count> ::= 0:#b11111111

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0007,#b10101010,0,0

Description: Because TTLTRG assertions are latched, it is not necessary that all of the specified lines be set at the same time; only that each specified line undergo an unasserted-to-asserted transition since the last TTLTRG Wait or TTLTRG Arm operation. Any bits set to 1 in bits 0-7 of <Count> represent corresponding TTLTRG lines which await assertion.

<Address> and <Misc> are not used.

See the related TTLTRG Control (0003) and TTLTRG Arm (0006) operations.

IRQ Arm 0008

Clears a set of latched IRQ assertions from a specified logical address.

Sequence Syntax: #h0008,<Count>,<Address>,<Misc>

<Count> ::= 0:255

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0008,86,0,0

Description: Clearing latched IRQ assertions guarantees that any subsequent IRQ Wait will not be satisfied by an old latched IRQ assertion. The <Count> field indicates the logical address for which the latched IRQ should be cleared.

<Address> and <Misc> are not used.

IRQ Wait 0009

Waits for IRQ from a specific logical address.

Sequence Syntax: #h0009,<Count>,<Address>,<Misc>

<Count> ::= 1:255

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h0009,222,0,0

Description: When a VXI system is configured, each IRQ line is assigned an IRQ Handler. The IRQ Handler may be any device which supports this capability. In order for the HP E2749 to proceed after executing the IRQ Wait Sequence operation, it must receive and IRQ from the specific logical address on any IRQ line for which it has been assigned as IRQ Handler. See the documentation for your VXI Resource Manager to determine how to assign IRQ Handlers.

<Address> and <Misc> are not used.

Pause N loops 000a

The Sequence stops executing for a designated number of loops.

Sequence Syntax: #h0002,<Count>,<Address>,<Misc>

<Count> ::= 1:4294967295

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h000a,10,0,0

Description: This operation causes the Sequence to execute a delay loop for the number of repetitions designated by <Count>. This operation may be used to pause a Sequence for a shorter duration of time than may be achieved with the 'Pause N milliseconds' (0002) operation for which the minimum time is 10 milliseconds. A <Count> of 1560927 results in a delay of ≥ 1 second. The actual delay time may be longer due to the unpredictable nature of interrupts.

<Address> and <Misc> are not used.

Lbus Consume 1000

A transmit operation which reads blocks of data from the local bus and transmits them to the Fibre Channel output.

Sequence Syntax: #h1000,<Count>,<Address>,<Misc>

<Count> ::= 1:256

<Address> ::= 0

<Misc> ::= 0:3 #h10:#hFFFF (see description below)

SCPI example: SEQ:ADD #h1000,8,0,#h03000800

Description: The Lbus Consume operation puts the local bus chip into a mode which acts as a sink for bytes on the local bus. In other words no bytes are passed to the next module to the right.

<Count> indicates the number of local bus blocks to transfer.

<Address> is not used.

<Misc> contains two pieces of information: the lower 24 bits indicate the number of bytes in a local bus block; the upper 8 bits indicate the local bus width. The value indicating the local bus width is presented as the number of bytes minus 1:

Bits 24-31 of Misc parameter		Bits 0-23 of Misc parameter
A local bus width of:	Is represented by a parameter value of:	This value represents the number of bytes in a local bus block. Every local bus block is assumed to be the same size and equal to the count specified here.
8	0	
16	1	
24	2	
32	3	

The bytes-per-block value is used to decrement the bytes-remaining count, thus determining when the final Sequence count has been met. The number of bytes per block must be specified correctly in order for the Sequence to terminate properly.

Lbus Eavesdrop 1001

A transmit operation which reads blocks of data from the local bus and transmits them to the Fibre Channel output in addition to passing them along to the next local bus module to the right.

Sequence Syntax: #h1001,<Count>,<Address>,<Misc>

<Count> ::= 1:256

<Address> ::= 0

<Misc> ::= 0:3 #h10:#hFFFF (see description below)

SCPI example: SEQ:ADD #h1001,2,0,#h3004000

Description: The Lbus Eavesdrop operation puts the local bus chip into a mode in which each byte received from the module to the left is transmitted through the HP E2749 to Fibre Channel and is also passed to the next module to the right.

<Count> indicates the number of local bus blocks to transfer.

<Address> is not used.

<Misc> contains two pieces of information: the lower 24 bits indicate the number of bytes in a local bus block; the upper 8 bits indicate the local bus width. The value indicating the local bus width is presented as the number of bytes minus 1:

Bits 24-31 of Misc parameter		Bits 0-23 of Misc parameter
A local bus width of:	Is represented by a parameter value of:	This value represents the number of bytes in a local bus block. Every local bus block is assumed to be the same size and equal to the count specified here.
8	0	
16	1	
24	2	
32	3	

The bytes-per-block value is used to decrement the bytes-remaining count, thus determining when the final Sequence count has been met. The number of bytes per block must be specified correctly in order for the Sequence to terminate properly.

Lbus Generate 2000

A receive operation which reads blocks of data from the Fibre Channel input then writes them to the local bus.

Sequence Syntax: #h2000,<Count>,<Address>,<Misc>

<Count> ::= 1:256

<Address> ::= 0

<Misc> ::= 0:3 #h10:#hFFFC (see description below)

SCPI example: SEQ:ADD #h2000,16,0,#h03000c00

Description: The Lbus Generate operation causes data to flow from the Fibre Channel input to the next module to the right of the HP E2749. This operation can only be used for local bus receive Sequences.

<Count> indicates the number of local bus blocks to transfer.

<Address> is not used.

<Misc> contains two pieces of information: the lower 24 bits indicate the number of bytes in a local bus block; the upper 8 bits indicate the local bus width. The value indicating the local bus width is presented as the number of bytes minus 1:

Bits 24-31 of Misc parameter		**Bits 0-23 bits of Misc parameter**
A local bus width of:	Is represented by a parameter value of:	This value represents the number of bytes in a local bus block and must be a multiple of 4. Every local bus block is assumed to be the same size and equal to the count specified here.
8	0	
16	1	
24	2	
32	3	

A block marker is asserted on the local bus following every block-size number of bytes. A frame marker is placed following the last block written to the local bus by this Sequence operation.

Lbus Append 2001

A receive operation which reads blocks of data from the Fibre Channel input then appends them to the local bus stream of blocks.

Sequence Syntax: #h2001,<Count>,<Address>,<Misc>

<Count> ::= 1:256

<Address> ::= 0

<Misc> ::= 0:3 #h10:#hFFFC (see description below)

SCPI example: SEQ:ADD #h2001,4,0,#h03000800

Description: The Lbus Append operation causes data to flow from the Fibre Channel input and appends the data to the end of an Lbus frame as it passes to the next module to the right of the HP E2749. This operation can only be used for local bus receive Sequences.

<Count> indicates the number of local bus blocks to transfer.

<Address> is not used.

<Misc> contains two pieces of information: the lower 24 bits indicate the number of bytes in a local bus block; the upper 8 bits indicate the local bus width. The value indicating the local bus width is presented as the number of bytes minus 1:

Bits 24-31 of Misc parameter		Bits 0-23 of Misc parameter
A local bus width of:	Is represented by a parameter value of:	This value represents the number of bytes in a local bus block and must be a multiple of 4. Every local bus block is assumed to be the same size and equal to the count specified here.
8	0	
16	1	
24	2	
32	3	

A block marker is asserted on the local bus following every block-size number of bytes. A frame marker is placed following the last block written to the local bus by this Sequence operation.

Read A16 Buff 16 – Read Shared RAM

3000-3012

Transmit operations which read data from a memory buffer or FIFO and transmit to the Fibre Channel output.

Sequence Syntax: #h3000,<Count>,<Address>,<Misc>
through
#h3012,<Count>,<Address>,<Misc>

<Count> ::= 4:#hFFFFFFFC

A16 <Address> ::= 0:#hFFFE

A24 <Address> ::= 0:#hFFFFFE

A32 <Address> ::= 0:#hFFFFFFFE

Shared RAM <Address> ::= 0:262144

<Misc> ::= 0

SCPI example: SEQ:ADD #h300B,#h10000,#hD420,0

Notes: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 19 operations included in this description.

Description: This description covers 19 operations for which some essential properties are indicated in the operation name. Buff indicates a memory buffer whereas FIFO refers to reading from the same address as if reading from a FIFO. The buffer or FIFO corresponds to the address space specified in the name: A16, A24, A32, or Shared RAM. The last part of the operation name refers to an access type: a 16-bit access, a 32-bit access implemented as two 16-bit accesses, or a D32 access. The D32 access applies only to devices which support D32. Shared RAM is always accessed as a 16-bit buffer.

<Count> designates the number of bytes to transfer and must be a multiple of 4.

<Address> designates an offset in the specified memory space (A16, A24, A32, or Shared RAM) at which memory will be accessed. The value must be a multiple of 2.

<Misc> is not used.

Transmit Dummy Bytes 3100

A transmit operation which places dummy bytes in the data stream.

Sequence Syntax: #h3100,<Count>,<Address>,<Misc>

<Count> ::= 0:#hFFFFFFFC

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h3100,#h10000,0,0

Description: This operation is used to add padding to certain data structures in the data stream to make it compatible with some post-processing programs which expect data in a certain location.

<Count> designates the number of dummy bytes to place in the data stream and must be a multiple of 4.

<Address> and <Misc> fields are not used.

Read Shared RAM Monitor Shared RAM - Read A24 Buff D32 Monitor A24 Buff

3812-3a05

Transmit operations which read data from the VXI bus and transmit to the Fibre Channel output while providing a means for the host computer to monitor the data.

Sequence Syntax: #h3812,<Count>,<Address>,<Misc>
through
#h3a05,<Count>,<Address>,<Misc>

<Count> ::= 4:#hFFFFFFFF

A16 <Address> ::= 0:#hFFFF

A24 <Address> ::= 0:#hFFFFFF

Shared RAM <Address> ::= 0:262144

A24 <Misc> ::= 0:#hFFFFFF

Shared RAM <Misc> ::= 0:262144

SCPI example: SEQ:ADD #h3a00,#h200,#h400,#h8000

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 27 operations included in this description.

Description: This description covers 27 operations for which some essential properties are indicated in the operation name:

The address location indicated in the operation name before the word 'Monitor' represents the memory location from which to draw data. The address location indicated in the operation name after the word 'Monitor' represents the memory location to which to monitor data.

Buff indicates a memory buffer whereas FIFO refers to reading from the same address as if reading from a FIFO. The buffer or FIFO corresponds to the address space specified in the name: A16, A24, A32, or Shared RAM.

<Count> is the number of bytes to monitor

<Address> is the VXI address from which to read data.

<Misc> is the VXI address to which to monitor data

Write A16 Buff 16 – Write Shared RAM

4000-4012

Receive operations which receive data from the Fibre Channel input and write to a memory buffer or FIFO.

Sequence Syntax: #h4000,<Count>,<Address>,<Misc>
through
#h4012,<Count>,<Address>,<Misc>

<Count> ::= 4:#hFFFFFFFC

A16 <Address> ::= 0:#hFFFE

A24 <Address> ::= 0:#hFFFFFE

A32 <Address> ::= 0:#hFFFFFFFE

Shared RAM <Address> ::= 0:262144

<Misc> ::= 0

SCPI example: SEQ:ADD #h400B,#h10000,#hD420,0

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 19 operations included in this description.

Description: This description covers 19 operations for which some essential properties are indicated in the operation name. Buff indicates a memory buffer whereas FIFO refers to writing to the same address as if writing to a FIFO. The buffer or FIFO corresponds to the address space specified in the name: A16, A24, A32, or Shared RAM. The last part of the operation name refers to an access type: a 16 bit access, a 32-bit access implemented as two 16-bit accesses, or a D32 access. Shared RAM is always accessed as a 16-bit buffer.

<Count> designates the number of bytes to transfer and must be a multiple of 4.

<Address> indicates an offset in the specified memory space (A16, A24, A32, or Shared RAM) at which memory will be accessed. This value must be a multiple of 2.

<Misc> is not used.

Bit Bucket 4100

A receive operation which discards bytes from the data stream.

Sequence Syntax: #h4100,<Count>,<Address>,<Misc>

<Count> ::= 4:#hFFFFFFFC

<Address> ::= 0

<Misc> ::= 0

SCPI example: SEQ:ADD #h4100,#h10000,0,0

Description: This operation can be used to receive a single channel from a multiple-channel transmit.

<Count> designates the number bytes to discard and must be a multiple of 4.

<Address> and <Misc> are not used.

Lbus Consume Monitor Shared RAM – Lbus Eavesdrop Monitor A24

5000-5015

Transmit operations which read data from the local bus and transmit to the Fibre Channel output while providing a means for the host computer to monitor the data via the VXI system bus.

Sequence Syntax: #h5000,<Count>,<Address>,<Misc>
through
#h5015,<Count>,<Address>,<Misc>

Monitor <Count> ::= 1:256

Pipe Monitor <Count> ::= 1:256 1:256 (see description below)

A24 <Address> ::= 0:#hFFFFFF

Shared RAM <Address> ::= 0:262144

<Misc> ::= 0:3 #h10:#hFFFF (see description below)

SCPI example: SEQ:ADD #h5016,#h100004,#h400000,#h3008000

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 4 operations included in this description.

Description: This description covers 4 operations for which some essential properties are indicated in the operation name:

- The part of the name preceding 'Monitor' indicates the type of transmit operation and the description corresponds to the description for the same type of operation described earlier in local bus transmit operations.

- In addition, the part of the name following 'Monitor' indicates the address space to which the data will be monitored (Monitor Shared RAM or Monitor A24).

The following considerations apply to transmit operations with a Monitor:

- All monitored data is passed into a memory buffer, not into a FIFO.

- All monitoring to Shared RAM is performed via D16 accesses.

- The A16 and A32 address spaces are not supported for monitoring.

- D32 monitoring is not available.

<Count> and <Misc> for monitoring are the same as the <Count> and <Misc> fields for the corresponding local bus transmit operations (1000-1003) described earlier.

<Address> indicates an offset in the specified memory space (A24 or Shared RAM). The memory block at the offset specified in the address space is 4 bytes larger than the local bus block size multiplied by the number of local bus blocks.

The following tips are applicable to running Sequences using Monitor:

- The act of monitoring a local bus transfer slows down the overall transmit rate because data must be copied to RAM which would not otherwise be done. The more data monitored, the slower the maximum transmit rate.
- You must specify all local bus blocks which are to be monitored before running the Sequence. Once the Sequence is running it is not possible to change which blocks to monitor.
- Flags are used to synchronize the host and the HP E2749 for monitor operations. The flags are represented by the first 4 bytes of the memory to which Monitor data is being written, beginning at the address specified in that memory (A24 or Shared RAM). You must initialize all the flag values before running the Sequence.

  The flag is used to indicate the presence of data in the Monitor block. When the flag is 0 the HP E2749 will write data into the block and set the flag to 1. It is expected that the host (or controller) will read the data and then set the flag to 0. If the monitor operation is executed with the flag non-zero, the memory copy will not be done, but the data will flow through the normal data stream to the SCSI Session. This allows the host to read data at a different rate than the actual acquisition of data without affecting the transmit rate. In fact transmits will be faster when the flag is set because the memory copy will not need to be done.

  The intention for monitoring many channels is that there will be a Sequence operation to monitor one block for each of the many channels. The flag values will initially be set to non-zero which means that no data will be copied to memory. As you decide to monitor different local bus blocks, the flag can be cleared allowing data to be written to that monitor block. Upon seeing the 0 flag the HP E2749 will write data to that block and then set the flag indicating that a block of data is available. This scheme will allow you to change which local bus blocks are being monitored during the transmit.

Wait Bit Set A16 – Wait Bit Clear Shared RAM

6000-6007

Synchronization operations which can be used to wait for data to be available from a device which generates data slower than the HP E2749 can transfer it.

Sequence Syntax: #h6000,<Count>,<Address>,<Misc>
through
#h6007,<Count>,<Address>,<Misc>

<Count> ::= 0:#b1111111111111111

A16 <Address> ::= 0:#hFFFF

A24 <Address> ::= 0:#hFFFFFF

A32 <Address> ::= 0:#hFFFFFFFF

Shared RAM <Address> ::= 0:262144

<Misc> ::= 0:#hFFFFFFFF

SCPI example: SEQ:ADD #h6002,0,#h380024,#b1000

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 8 operations included in this description.

Description: These operations wait for a single bit or a group of bits to be set or cleared in another device. This description covers 8 operations for which some essential properties are indicated in the operation name. Each reference is to a 16 bit value and performs a D16 access to the memory location specified. Both the memory space referenced and whether to wait for the bit(s) to be set or cleared are indicated in the name of the operation.

<Count> specifies a bit mask which is ANDed with the 16 bit register specified by the memory offset and memory space. For the Set operations all bits in the mask must be set. For the Clear operations all bits in the mask must be clear. The HP E2749 reads this register and checks the bits until the condition is met. If the condition is never met the Sequence will not be completed.

<Address> indicates the offset into the memory space indicated in the operation name.

<Misc> represents a user-programmable delay prior to the next VXI access. The number of loops specified here is performed before another VXI access. This frees the VXI bus to perform additional activities, rather than having Sequence operations completely dominate VXI bus usage. Loop time is approximately 3 μs.

Wait A16 Count16– Wait Count Shared RAM 32

6008-600f

Synchronization operations which can be used to wait for data to be available from a device which generates data slower than the HP E2749 can transfer it.

Sequence Syntax: #h6008,<Count>,<Address>,<Misc>
through
#h600f,<Count>,<Address>,<Misc>

16 bit <Count> ::= 1:#hFFFF

32 bit <Count> ::= 1:#hFFFFFFFF

A16 <Address> ::= 0:#hFFFF

A24 <Address> ::= 0:#hFFFFFF

A32 <Address> ::= 0:#hFFFFFFFF

Shared RAM <Address> ::= 0:262144

<Misc> ::= 0:#hFFFFFFFF

SCPI example: SEQ:ADD #h600B,#h4000,128000,0

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6)for a list of all 8 operations included in this description.

Description: These operations wait for the count register in another device to be greater than the value specified by <Count>. This description covers 8 operations for which some essential properties are indicated in the operation name. The memory space referenced is indicated in the name of the operation (A16, A24 or A32, or Shared RAM). The count register may be a 16-bit or a 32-bit value as indicated by the name of the operation, but all accesses are done using D16 (a 32-bit count will be performed by using two 16-bit accesses).

<Count> specifies the number which must be met or exceeded before proceeding. The HP E2749 reads this register and checks the count until the condition is met. If the condition is never met the Sequence will not be completed.

<Address> indicates the offset into the memory space indicated in the operation name.

<Misc> represents a user-programmable delay prior to the next VXI access. The number of loops specified here is performed before another VXI access. This frees the VXI bus to perform additional activities, rather than having Sequence operations completely dominate VXI bus usage. Loop time is approximately 3 μs.

Control A16 Reg 16– Control Reg Shared RAM 32

6018-601f

Synchronization operations which allow the HP E2749 to write directly to a memory location.

Sequence Syntax: #h6018,<Count>,<Address>,<Misc>
through
#h601F,<Count>,<Address>,<Misc>

<Count> ::= 0

A16 <Address> ::= 0:#hFFFF

A24 <Address> ::= 0:#hFFFFFF

A32 <Address> ::= 0:#hFFFFFFFF

Shared RAM <Address> ::= 0:262144

<Misc> ::= #h0:FFFFFFFF

SCPI example: SEQ:ADD #h601A,0,#h2D860860,#hF020

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 8 operations included in this description.

Description: These operations allow the HP E2749 to write to a memory location, usually for the purpose of controlling another device on the bus. The register can be either 16 bits wide or 32 bits wide (a 32-bit register is written as two D16-bit writes). The memory space and register width are indicated in the Sequence name.

<Count> is not used.

<Address> indicates the offset into the memory space indicated in the operation name.

<Misc> contains the value which is to be written to the memory location specified by the memory space and the memory offset in the <Address> field.

Dump A24 Seq Bytes- Dump Shared RAM Seq Bytes

6020-6022

Write to a memory space the number of bytes which have been transferred.

Sequence Syntax:

#h6020,<Count>,<Address>,<Misc>
through
#h6022,<Count>,<Address>,<Misc>

<Count> ::= 0

A24 <Address> ::= 0:#hFFFFFF

A32 <Address> ::= 0:#hFFFFFFFF

Shared RAM <Address> ::= 0:262144

<Misc> ::= 0

SCPI example: SEQ:ADD #h6022,0,#h100,0

Note: Refer to the Sequence Quick Reference part of this chapter (beginning on page 7-6) for a list of all 3 operations included in this description.

Description: During both transmit and receive Sequences an internal counter keeps a count of how many bytes have been transferred. The contents of this counter may be written to a memory space in order to monitor progress of a Sequence.

<Address> indicates the offset into the memory space indicated in the operation name.

<Count> and <Misc> are not used.

Test Shared RAM and Skip 7000

Execute the next sequence operation only if a shared RAM location is non-zero.

Sequence Syntax: #h7000,<Count>,<Address>,<Misc>

<Count> ::= 0

Shared RAM <Address> ::= 0:262142

<Misc> ::= 0:100

SCPI example: SEQ:ADD #h7000,0,0,1

Description: Read a 16-bit value at the specified shared RAM <address>. If the value read is zero, skip the next <misc> number of sequence operations. If the value read is non-zero, set the 16-bit value in shared RAM to zero and execute the next sequence operation.

8

Programming the HP E2749 with SCPI

SCPI (Standard Commands for Programmable Instruments) is an industry-standard instrument control language. SCPI builds on the IEEE 488.1 and 488.2 standards.

Getting started

SCPI command structure and format

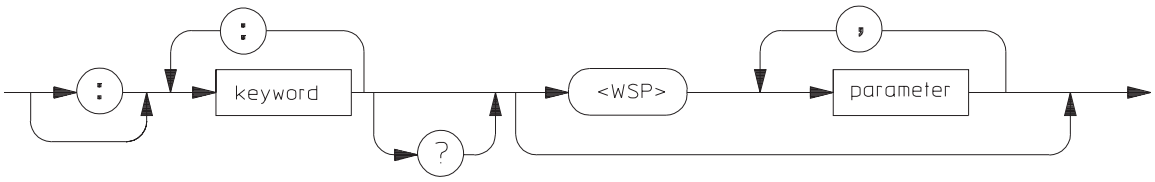
SCPI organizes related functions by grouping them together on a common branch of a command tree. Each branch is assigned a keyword to indicate the nature of the related functions. For example, the functions that control and monitor the status registers are grouped under the STATUS branch of the command tree. The STATUS branch is only one of the major SCPI branches which are called subsystems.

Colons indicate branching points on the command tree. A parameter is separated from the rest of the command by a space.

You can send multiple commands within a single message by separating commands with semicolons. One of the main functions of the command parser is to keep track of a program message's position in the command tree. If a program message contains two commands separated by a semicolon, the command parser assumes that the keywords of the second command come from the same branch of the tree as the final keyword of the preceding command. This allows you to simplify multiple command program messages.

Another way to simplify program messages is to delete implied mnemonics. You can omit some keywords from the command without changing the effect of the command. Implied mnemonics are identified by brackets [ ] in SCPI syntax diagrams.

The illustration below describes the basic syntax of SCPI commands.



NOTE:
WSP = whitespace, ASCII character (Decimal 0-9 or 11-32)

For additional information about SCPI command structure and format, see the *Beginner's Guide to SCPI*, available through your local Hewlett-Packard Sales Office.

Parameter settings

As the illustration shows, there must be a <WSP>, whitespace or <space>, between the last command keyword and the first parameter in a command. This is one of the few places in SCPI where <space> is required. If you send more than one parameter with a single command, you must separate adjacent parameters using a comma.

Each parameter format has one or more corresponding response-data formats. For example, a setting that you program using a numeric parameter would return either floating point or integer response data when queried. Whether floating point or integer response data is returned, depends on the particular VXI module you are using. However, response data is clearly defined for the module and query. The next chapter, "SCPI Command Reference" specifies the data format for individual commands.

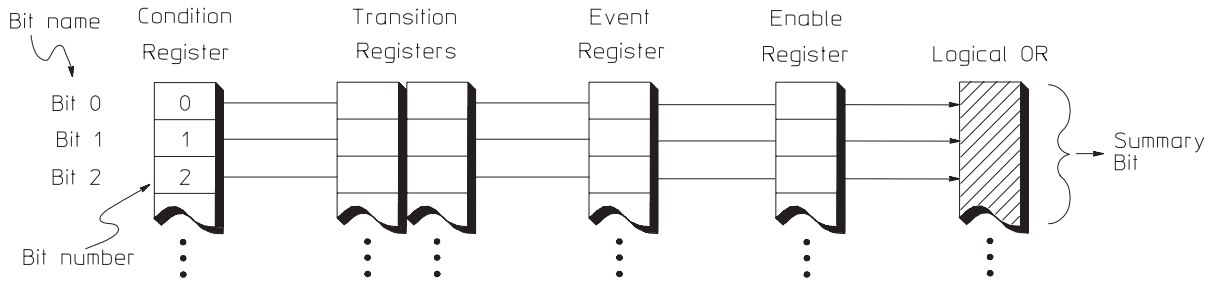
For additional information about SCPI data formats, see the *Beginner's Guide to SCPI*, available through your local Hewlett-Packard Sales Office.

Using the status registers

The HP E2749's status registers contain information about various module conditions. The following sections describe the registers and tell you how to use them in your programs.

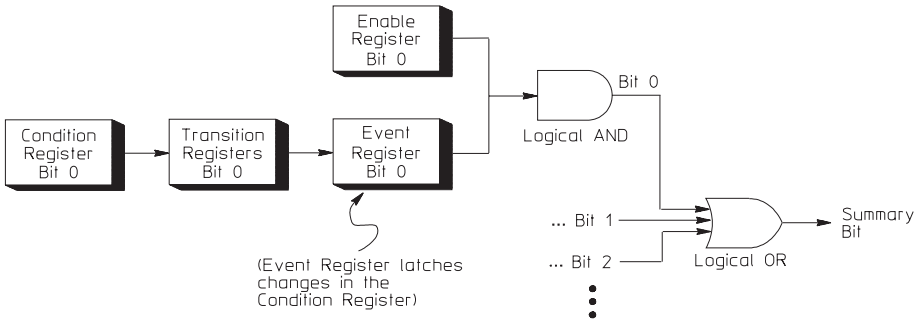
The general status register model

The general status register model, shown below is the building block of the HP E2749's status system. Most register sets in the module include all of the registers shown in the general model, although commands are not always available for reading or writing a particular register. The model consists of a condition register, two transition registers, an event register, and an enable register.



The flow within a status group starts at the condition register and ends at the register summary bit. (See the illustration below.) You control the flow by altering bits in the enable and transition registers.

The Operation Status and Questionable Status groups are 16 bits wide, while the Status Byte and Standard Event groups are 8 bits wide. In the 16-bit groups, the most significant bit (bit 15) is not used. Bit 15 is always set to 0.



Condition register

The condition register continuously monitors hardware and firmware status. It represents the current state of the module. It is updated in real time. When the condition monitored by a particular bit becomes true, the bit is set to 1. When the condition becomes false, the bit is reset to 0. Condition registers are read-only.

If there is no command to read a particular condition register, it is simply invisible to you.

The Transition registers

The positive and negative transition registers specify which type of bit transition in the Condition register will set corresponding bits in the Event register. Transition register bits may be set for positive transitions (0 to 1), or negative transitions (1 to 0).

Each bit set in the negative transition register indicates that a 1 to 0 transition of that bit in the Condition register sets the associated bit in the Event register. Each bit set in the positive transition register indicates that a 0 to 1 transition of that bit in the Condition register sets the associated bit in the Event register. Setting the same bits in both the positive and negative transition registers indicates that any transition of those bits in the Condition register sets corresponding bits in the Event register.

Event register

The event register records condition changes. When a change occurs in the condition register, the corresponding event bit is set to 1 in accordance with the transition register settings. Once set, an event bit is no longer affected by condition changes and subsequent events corresponding to that bit are ignored. The event bit remains set until the event register is cleared— either when the register is read or when the *CLS (clear status) command is sent. Event registers are read-only.

Note Reading the event register, clears the event register.

Enable register

The enable register specifies which bits in the event register set a summary bit to 1. The module logically ANDs corresponding bits in the event and enable registers, and ORs all the resulting bits to determine the state of a summary bit. Summary bits are in turn recorded in the Status Byte. (The summary bit is only set to 1 if one or more enabled event bits are set to 1.) Enable registers are read-write.

Enable registers are cleared by *CLS (clear status). Querying enable registers does not affect them. There is always a command to read and write to the enable register of a particular register set.

How to use registers

There are two methods you can use to access the information in register sets:

- The polling method
- The service request (SRQ) method

Use the polling method when:

- Your language/development environment does not support SRQ interrupts.
- You want to write a simple, single-purpose program and do not want to add the complexity of setting up an SRQ handler.

Use the SRQ method when:

- You need time-critical notification of changes.
- You are monitoring more than one device which supports SRQ.
- You need to have the controller do something else while it is waiting.
- You cannot afford the performance penalty inherent to polling.

The Polling method

In the polling method, the module has a passive role. It only tells the controller that conditions have changed when the controller asks the right question. In the SRQ method, the module notifies the controller of a condition change without the controller asking. Either method allows you to monitor one or more conditions.

When you monitor a condition with the polling method, you must

1 Determine which register contains the bit that monitors the condition.

2 Send the unique SCPI query that reads that register.

3 Examine the bit to see if the condition has changed.

The polling method works well if you do not need to know about changes the moment they occur. The SRQ method is more effective if you must know immediately when a condition changes. To detect a change in a condition using the polling method, your program would need to continuously read the registers at very short intervals. This makes the program less efficient. In this case it is better to use the SRQ method.

The SRQ method

When you monitor a condition with the SRQ method, you must

1 Determine which bit monitors the condition.

2 Determine how that bit reports to the request service (RQS) bit of the Status Byte.

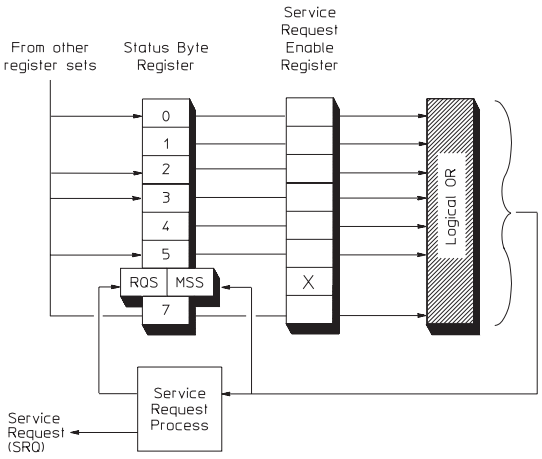
3 Send SCPI commands to enable the bit that monitors the condition and to enable the summary bits that report the condition to the RQS bit.

4 Enable the controller to respond to service requests.

When the condition changes, the module sets its RQS bit and generates an SRQ. The controller is informed of the change as soon as it occurs. The time the controller would otherwise have used to monitor the condition can now be used to perform other tasks. Your program determines how the controller responds to the SRQ.

Generating a service request

To use the SRQ method, you must understand how service requests are generated. As shown below, other register sets in the module report to the Status Byte. Many of them report directly, but some may report indirectly.



Bit 6 of the Status Byte serves two functions; the request service function (RQS) and the master summary status function (MSS). The RQS bit changes whenever something changes that it is configured to report. The RQS bit is cleared when it is read with a serial poll. The MSS bit is set in the same way as the RQS bit. However, the MSS bit is cleared only when the condition that set it is cleared. The MSS bit is read with *STB?.

When a register set causes its summary bit in the Status Byte to change from 0 to 1, the module can initiate the service request (SRQ) process. However, the process is only initiated if both of the following conditions are true:

- The corresponding bit of the Service Request enable register is also set to 1.
- The module does not have a service request pending.
- (A service request is considered to be pending between the time the module's SRQ process is initiated and the time the controller reads the Status Byte register with a serial poll.)

The SRQ process generates an SRQ. It also sets the Status Byte's request service (RQS) bit to 1. Both actions are necessary to inform the controller the module requires service. Generating an SRQ only informs the controller that some device on the bus requires service. Setting the RQS bit allows the controller to determine which device requires service. That is, it tells the controller that this particular device requires service.

If your program enables the controller to detect and respond to service requests, it should instruct the controller to perform a serial poll of all modules when an SRQ is generated. Each device on the bus returns the contents of its Status Byte register in response to this poll. The device whose RQS bit is set to 1 is the device that requested service.

Note

When you read the module's Status Byte with a serial poll, the RQS bit is reset to 0. Other bits in the register are not affected.

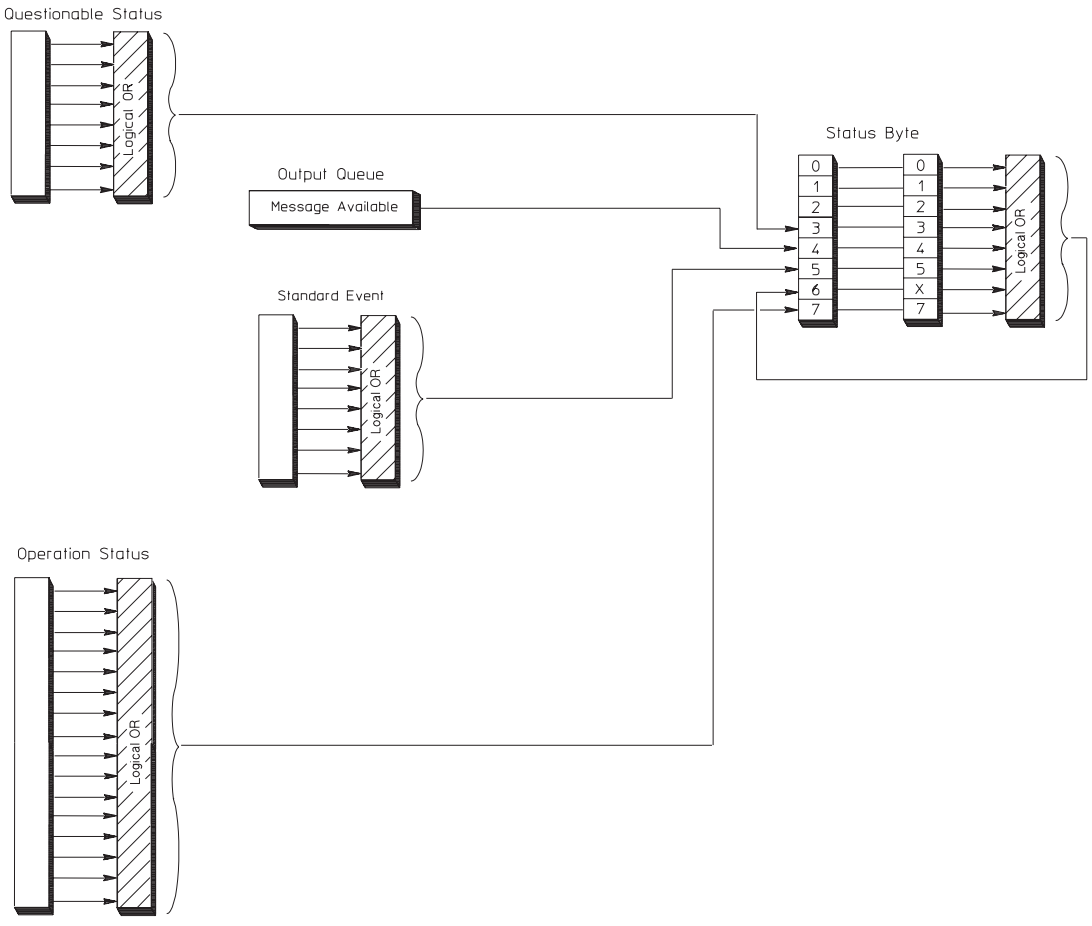
The HP E2749 registers sets

The HP E2749 uses four register sets to keep track of the module's status:

- Status Byte
- Questionable Status
- Standard Event
- Operational Status

Their reporting structure is summarized in the illustration below. They are described in greater detail in the following sections.

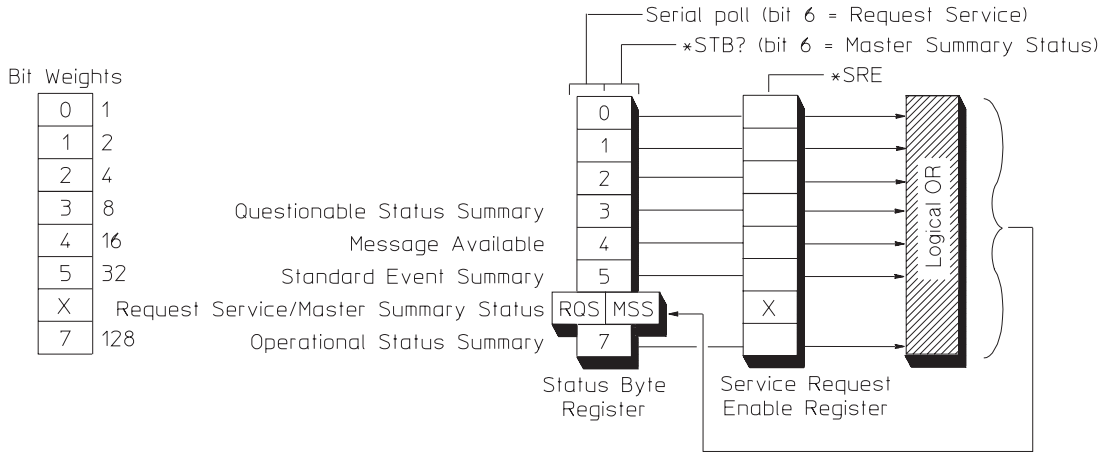
Register bits not explicitly presented in the following sections are not used in the HP E2749. A query to one of these bits returns a value of 0.



HP E2749 Register Sets

Status byte

The Status Byte summarizes the states of the other register sets and monitors the HP E2749's output queue. It is also responsible for generating service requests (see "Generating Service Requests" earlier in this chapter).



The Status Byte is unique because it does not exactly conform to the general status model presented earlier. It contains only two registers: the Status Byte register and the Service Request enable register. The Status Byte registers behaves like a condition register for all bits except bit 6. The Service Request enable behaves like a standard enable register except that bit 6 is always set to 0.

Bits in the Status Byte register are set to 1 under the following conditions:

- Questionable Status Summary (bit 3) is set to 1 when one or more enabled bits in the Questionable Status event register are set to 1.
- Message Available (bit 4) is set to 1 when the output queue contains a response message.
- Standard Event Summary (bit 5) is set to 1 when one or more enabled bits in the Standard Event event register are set to 1.
- Master Summary Status (bit 6, when read by *STB?) is set to 1 when one or more enabled bits in the Status Byte register are set to 1.
- Request Service (bit 6, when read by serial poll) is set to 1 by the service request process (see "Generating a Service Request" earlier in this chapter).
- Operation Status Summary (bit 7) is set to 1 when one or more enabled bits in the Operation Status event register are set to 1.

The illustration also shows the commands you use to read and write the Status Byte registers. The following statements are example commands using the Status Byte and Status Byte enable register.

***SRE 16** Generate an SRQ interrupt when messages are available in the output queue.

***SRE?** Find out what events are enabled to generated SRQ interrupts.

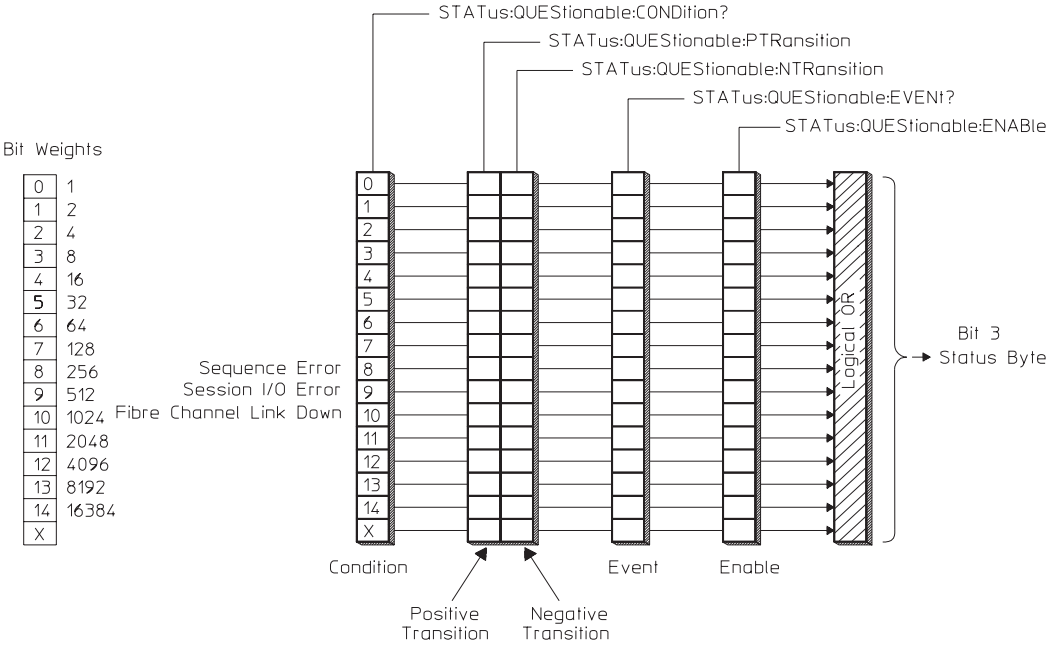
***STB?** Read the Status Byte event register.

See "Setting and Querying Registers" later in this chapter for more information about these commands.

Questionable status register set

The Questionable Status register monitors conditions that affect the quality of the data transfer.

This register set includes a condition register, two transition registers, an event register, and an enable register. It is accessed through the STATUS subsystem. See "Setting and Querying Registers" later in this chapter for more information about using these commands.



The condition register

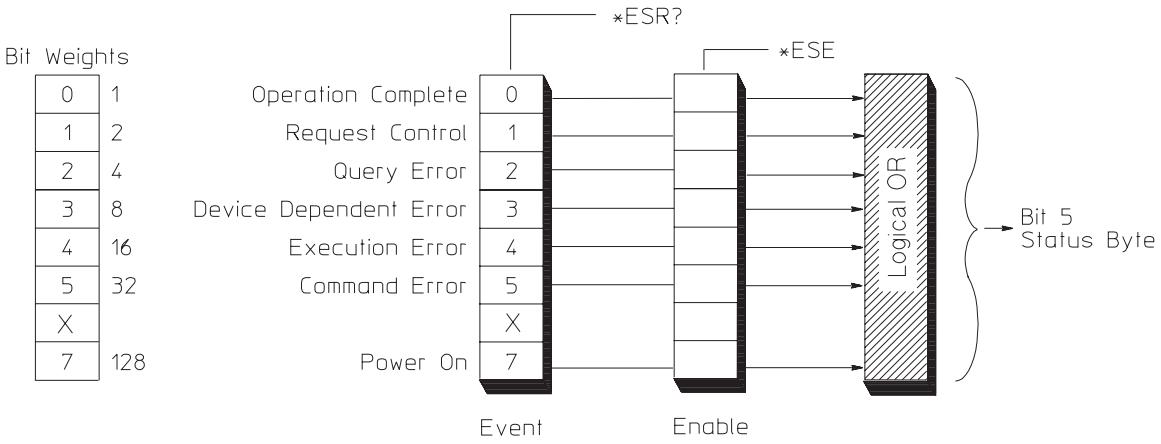
Bits in the Questionable Status condition register are set to 1 under the following conditions:

- Sequence Error (bit 8) is set to 1 when an error is detected during Sequence execution
- Session I/O Error (bit 9) is set to 1 when an error is detected during Session I/O operation

The illustration shows the commands you use to read and write the Questionable Status registers.

Standard event status register set

The Standard Event Status register set monitors module errors as shown below. It is one of the simplest and most frequently used. The unique aspect of this group is that you program it using common commands, while you program other register sets through the STATUS subsystem.



The Standard Event Status Register set does not conform to the general status register model described at the beginning of this chapter. It contains only two registers: the Standard Event Status event register and the Standard Event Status enable register.

Bits in the Standard Event Status event register are set to 1 under the following conditions:

- Operation Complete (bit 0) is set to one when the following two events occur (in the order listed):

– You send the *OPC command to the module.

– The module completes all pending overlapped commands.

- Query Error (bit 2) is set to 1 when the module detects a query error.
- Device Dependent Error (bit 3) is set to 1 when the command parser or execution routines detect a device-dependent error.
- Execution Error (bit 4) is set to 1 when the command parser or execution routines detect an execution error.
- Command Error (bit 5) is set to 1 when the command parser detects a command or syntax error.
- Power On (bit 7) is set to 1 when you turn on the module.

The illustration also shows the commands you use to read and write the Standard Event Status register sets. Example commands using Standard Event Status registers:

***ESE 48** Generate a summary bit whenever there is an execution or command error

***ESE?** Query the state of the Standard Event Status enable register?

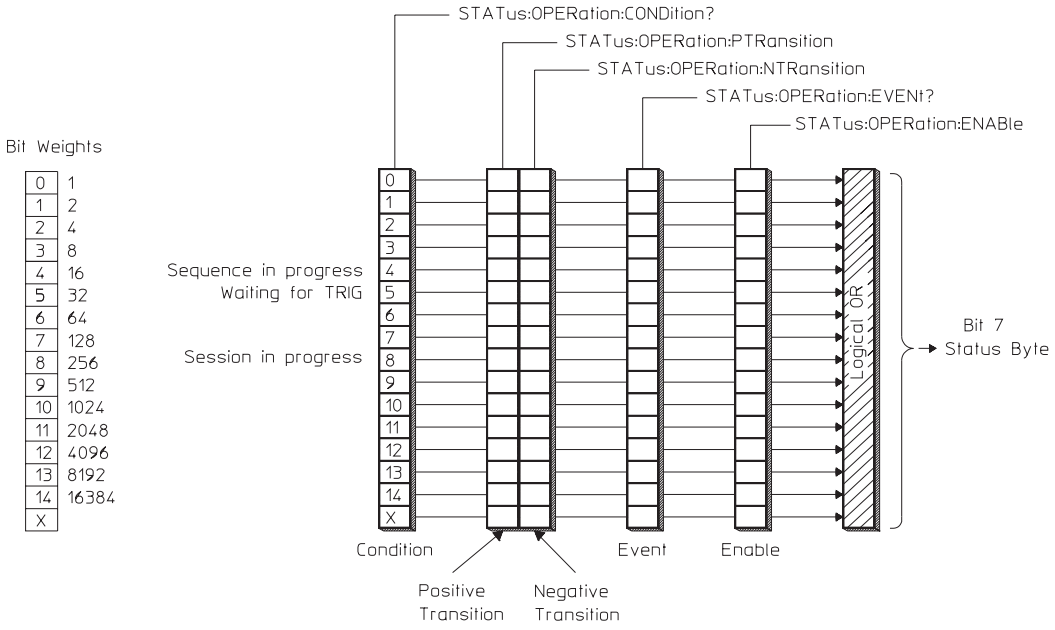
***ESR?** Query the state of the Standard Event Status event register.

See “Setting and Querying Registers” later in this chapter for more information about using these commands.

Operation status register set

The Operation Status register set monitors conditions in the module's data transfer process.

This register set includes a condition register, two transition registers, an event register, and an enable register. It is accessed through the STATUS subsystem. See "Setting and Querying Registers" later in this chapter for more information about using these commands.



Bits in the Operation Status condition register are set to 1 under the following conditions:

- Sequence in Progress (bit 4) is set to 1 while a Sequence is in progress and to 0 when the Sequence has finished.
- Waiting for TRIG (bit 5) is set to 1 when the module is ready to accept a trigger signal from one of the trigger sources. (If a trigger signal is sent before this bit is set, the signal is ignored.)
- Session in Progress (bit 8) is set to 1 while a Session is in progress and to 0 when a Session is has finished.

The illustration shows the commands you use to read and write the Operation Status registers.

Setting and Querying Registers

The previous register set illustrations include the commands you use to read from and to write to the registers. Most commands have a *set form* and a *query form*.

Use the set form of the command to write to a register. The set form is shown in the illustrations. The set form of a command takes an extended numeric parameter.

Use the query form of the command to read a register. Add a "?" to the set form to create the query form of the command. Commands ending with a "?" in the illustrations are query-only commands. These commands cannot set the bits in the register, they can only query or read the register.

The register set illustrations also include the bit weights you use to specify each bit in the register. For example, to get the Waiting for Trigger condition register (bit 5 in Operation Status register set) to generate a service request, send the following commands:

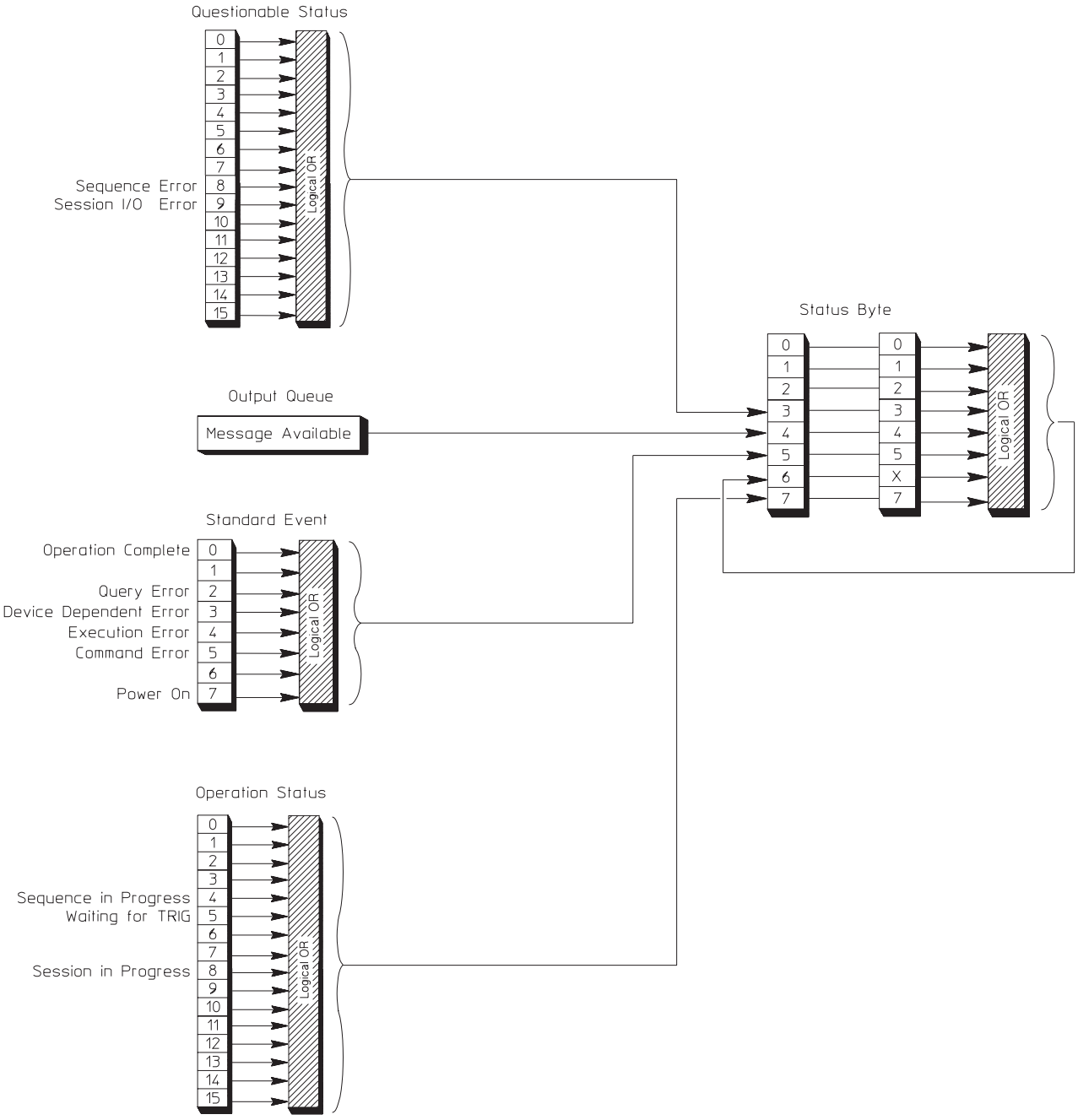
STATUS:PRESET Sets the Enable register bits in the Operational Status and the Questionable Status register sets to 0.

STATUS:OPERATION:ENABLE 32 Sets the Waiting for Trigger Enable register (bit 5) to 1.

***SRE 128** Sets bit 7 of the Service Request Enable register to 1.

See the next chapter for more information about these commands.

HP E2749 Register Set Summary



Using SCPI commands in your environment

SCPI commands do not require special drivers or downloadable files. Your controller can simply send SCPI commands to and read ASCII-formatted responses from the HP E2749 module.

The application program running on your controller may be written in HP VEE, HP BASIC, C, or some other programming language. Documentation provided with your controller and its programming language will show how to send SCPI commands to and read responses from the HP E2749 module.

9

SCPI Command Reference

SCPI Command Fields

The Command Reference chapter describes all of the HP E2749 's SCPI commands. Each command has the following:

1 The heading. This includes two fields. The field to the left shows the command name. The field to the right indicates whether the command has a command form, a query form, or both.

2 A brief description of the command. This one- or two-line description appears just below the heading.

3 A syntax description. This may consist of one or two parts: only a command syntax, only a query syntax, or both. The syntax description shows you the syntax expected by the command parser. A detailed description for the elements appearing in the syntax description follows. For additional information about message syntax see the *Beginner's Guide to SCPI*, available through your local Hewlett-Packard Sales Office.

4 Example query/command. This field appears at the end of the syntax description for some commands. It contains one or two examples of the command or query.

5 A return format description. This field is only used if the command has a query form. It tells you how data is returned in response to the query.

6 An attribute summary. This field defines the command's preset state, identifies overlapped commands requiring synchronization, and specifies compliance with SCPI. A "confirmed" command complies with SCPI 1994.

7 A detailed description. This field contains additional information about the command.

Finding the Right Command

- If you do not find a command where you expect it, try scanning the quick reference tables that begin on page 9-7 for the equivalent command that contains the implied mnemonic.

  Each command has a brief description. After you locate the equivalent command, you can find a more detailed description in the command reference.

- If you are looking for a command that accesses a particular function, use the index.

  For example, if you want to find the command that resets the local bus, look for "reset local bus" in the index. It sends you to the pages that describe the VINStrument:LBUS:RESet command.

Command Syntax

This section describes the syntax elements used in the SCPI command reference. It also describes the general syntax rules for both kinds of command and query messages.

Special Syntactic Elements

Several syntactic elements have special meanings:

- colon (:) — When a command or query contains a series of keywords, the keywords are separated by colons. A colon immediately following a keyword tells the command parser that the program message is proceeding to the next level of the command tree. A colon immediately following a semicolon tells the command parser that the program message is returning to the base of the command tree.
- semicolon (;) — When a program message contains more than one command or query, a semicolon is used to separate them from each other. For example, if you want to have the HP E2749 interrupt the host when the sequence completes, the message would be:

«—Ꞧ–³¾–ff“ff¾○ff□ ½²ⁿ³›○“– ½²ⁿ³«–□› º½©

- comma (,) — A comma separates the data sent with a command or returned with a response. For example, the SEQuence:BEGin command requires three values to determine the destination, size, and data source of a Sequence which is to be executed. For example, a message to begin transmitting data to Fibre Channel using sequence 2 and Session 3 would be:

→□ᵛ₇⅝–⅛⅝ ½¾→□ⓓ– ff□“○ᴸꜰ№ⓓᴺ£©¼©©ⁿᵃ©£¼

- <WSP> — One white space is required to separate a program message (the command or query) from its parameters. For example, the command “SEQuence:BEGin VPL,262144,1” contains a space between the program header (SEQuence:BEGin) and its program data (VPL,262144,1). White space characters are not allowed within a program header.

Conventions

Syntax and return format descriptions use the following conventions:

- < > Angle brackets enclose the names of items that need further definition. The definition will be included in accompanying text. In addition, detailed descriptions of these elements appear at the end of this section.
- = "is defined as" When two items are separated by this symbol, the second item replaces the first in any statement that contains the first item. For example, A::=B indicates that B replaces A in any statement that contains A.
- | "or" When items in a list are separated by this symbol, one and only one of the items can be chosen from the list. For example, A|B indicates that A or B can be chosen, but not both.
- ... An ellipsis (trailing dots) is used to indicate that the preceding element may be repeated one or more times.
- [ ] Square brackets indicate that the enclosed items are optional.
- { } Braces are used to group items into a single syntactic element. They are most often used to enclose lists and to enclose elements that are followed by an ellipsis.

Although the command interpreter is not case sensitive, the case of letters in the command keyword is significant in the Command Reference. Keywords that are longer than four characters can have a short form or a long form. SCPI accepts either form. Upper-case letters show the short form of a command keyword.

SCPI is sensitive to white space characters. White space characters are not allowed within command keywords. They are only allowed when they are used to separate a command and a parameter.

A message terminator is required at the end of a program message or a response message. Use <NL>, <^END>, or <NL> <^END> as the *program message terminator*. The word <NL> is an ASCII new line (line feed) character. The word <^END> means that End or Identify (EOI) is asserted on the HP-IB interface at the same time the preceding data byte is sent. Most programming languages send these terminators automatically. For example, if you use the HP BASIC OUTPUT statement, <NL> is automatically sent after your last data byte. If you are using a PC, you can usually configure your system to send whatever terminator you specify.

Syntax Descriptions

Syntax descriptions in the SCPI command reference chapter use the following elements:

<CHAR> This item designates a string of ASCII characters. There are no delimiters. Usually, the string is from an explicit set of responses. Maximum length is 12 characters.

<STRING> This item specifies any 8-bit characters delimited by single quotes or double quotes. The beginning and ending delimiter must be the same. If the delimiter character is within the string, it must be entered twice. (For example, to get “EXAMPLE”, enter “”EXAMPLE"").

HP E2749 SCPI Quick Reference

Command	Description
Common Commands	
*CLS	Clears the Status Byte
*ESE	Sets or queries bits in the Standard Event Status enable register
*ESR?	Reads and clears the Standard Event Status event register
*IDN?	Returns module's identification string
*OPC	Enables status bit or query completion of all pending overlapped commands
*RST	Executes a device reset
*SRE	Sets or queries bits in the Service Request enable register
*STB?	Reads the Status Byte register
*TST?	Performs selftest
*WAI	Wait-to-continue command
Local Bus Configuration	
VINStrument[:CONFigure]:LBUS[:MODE]	Configures the local bus
VINStrument:LBUS:RESet	Resets the HP E2749 local bus
Session Control	
MMEMory:SESSion[1 | 2 | ... | 12]:PROTocol	Selects the communication protocol to be used for the specified session
Sequence Operations	
SEQuence[1 | 2 | 3 | 4]:ADD	Appends an operation to the specified Sequence
SEQuence[1 | 2 | 3 | 4]:BEGin	Begins a Sequence for data transfer
SEQuence[1 | 2 | 3 | 4]:DELete:ALL	Removes all operations from the specified Sequence list
SEQuence[1 | 2 | 3 | 4]:SIZE?	Returns the number of elements in the Sequence
SEQuence[1 | 2 | 3 | 4]:TRANsferred?	Returns the number of bytes transferred in the Sequence
Status Reporting	

	Command	Description
	STATus:OPERation:CONDition?	Reads the Operation Status condition register
	STATus:OPERation:ENABle	Sets and queries bits in the Operation Status enable register
	STATus:OPERation[:EVENt]?	Reads and clears the Operation Status event register
	STATus:OPERation:NTRansition	Sets and queries bits in the Operation Status negative transition register
	STATus:OPERation:PTRansition	Sets and queries bits in the Operation Status positive transition register
	STATus:PRESet	Sets bits in most enable and transition registers to the default state
	STATus:QUEStionable:CONDition?	Reads the Questionable Status condition register
	STATus:QUEStionable:ENABle	Sets and queries bits in the Questionable Status enable register
	STATus:QUEStionable[:EVENt]?	Reads and clears the Questionable Status event register
	STATus:Questionable:NTRansition	Sets and queries bits in the Questionable Status negative transition register
	STATus:Questionable:PTRansition	Sets and queries bits in the Questionable Status positive transition register
System Control		
	SYSTem:ABORt	Aborts a data transfer Session and/or Sequence
	SYSTem:COMMunicat:WWN[:SELF]:ADDRess	Stores the Fibre Channel WWN into ROM
	SYSTem:ERRor?	Returns one error message from the queue
	SYSTem:VERSion	Returns the SCPI version to which the module complies
Diagnostics		
	DIAGnostic:FIBRe:LOGin?	Returns logged in node information
	DIAGnostic:FIBRe:LOOPback	Performs a loopback test
	DIAGnostic:FIBRe:LOOPback:COUNt?	Returns the current loopback test iteration count
	DIAGnostic:FIBRe:LOOPback:MODe	Selects the loopback data path
	DIAGnostic:FIBRe:LWSYstran:VERSion?	Return the revision number of the Systran driver.
	DIAGnostic:LBUS:CONsume?	Tests the local bus data transfer to module
	DIAGnostic:LBUS:GENerate?	Tests the local bus data transfer from module

HP E2749 SCPI Commands

The following pages contain detailed descriptions of each SCPI command.

*CLS command

Clears the Status Byte by emptying the error queue and clearing all event registers.

Command Syntax: *CLS

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This command clears the Status Byte register. It does so by emptying the error queue and clearing (setting to 0) all bits in the event registers of the following register sets:

- Questionable Status
- Standard Event
- Operation Status

In addition, *CLS cancels any preceding *OPC command or query. This ensures that bit 0 of the Standard Event register will not be set to 1 and that a response will not be placed in the instrument's output queue when pending overlapped commands are completed.

*CLS does not change the current state of enable registers or transition filters.

Note To guarantee that the Status Byte's Message Available and Master Summary Status bits are cleared, send *CLS immediately following a Program Message Terminator.

For more information on the Status Byte register, see "The HP E2749's Register Sets" in the previous chapter.

*ESE command/query

Sets or queries bits in the Standard Event Status enable register.

Command Syntax: *ESE <Mask>

<Mask>::=number
limits: 0:255

Query Syntax: *ESE?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This command allows you to set bits in the Standard Event Status enable register. Assign a decimal weight to each bit you want set (to 1) according to the following formula:

$$2^{(\text{bit\_number})}$$

with acceptable values for bit_number being 0 through 7. Add the weights and then send the sum with this command.

When an enable register bit is set to 1, the corresponding bit of the Standard Event Status event register is enabled. All enabled bits are logically ORed to create the Standard Event Status summary, which reports to bit 5 of the Status Byte. Bit 5 is only set to 1 if both of the following are true:

- One or more bits in the Standard Event Status event register are set to 1.
- At least one set bit is enabled by a corresponding bit in the Standard Event Status enable register.

The query returns the current state of the Standard Event Status enable register. The state is returned as a sum of the decimal weights of all set bits.

For more information on the Standard Event Status register set, see "The HP E2749's Register Sets" in the previous chapter.

*ESR? query

Reads and clears the Standard Event Status event register.

Query Syntax: *ESR?

Return Format: Integer

Attribute Summary: Preset State: +0
Synchronization Required: no
SCPI Compliance: confirmed

Description: This query returns the current state of the Standard Event Status event register. The state is returned as a sum of the decimal weights of all set bits. The decimal weight for each bit is assigned according to the following formula:

$$2^{(\text{bit\_number})}$$

with acceptable values for bit_number being 0 through 7.

The query clears the register after it reads the register.

A bit in this register is set to 1 when the condition it monitors becomes true. A set bit remains set, regardless of further changes in the condition it monitors, until one of the following occurs:

- You read the register with this query.
- You clear all event registers with the *CLS command.

For more information on the Standard Event Status enable register set, see "The HP E2749's Register Sets" in the previous chapter.

*IDN? query

Returns a string that identifies the module.

Query Syntax: *IDN?

Return Format: HEWLETT-PACKARD,E2749,<serial_number>,<software_revision>

Attribute Summary: Preset State: instrument dependent
Synchronization Required: no
SCPI Compliance: confirmed

Description: This query returns:

- The name of the manufacturer, Hewlett Packard
- The product number, E2749A
- The serial number
- The version of the software

*OPC command/query

Enable status bit or query completion of all pending overlapped commands.

Command Syntax: *OPC

Query Syntax: *OPC?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: Some commands are processed sequentially. A sequential command holds off the processing of subsequent commands until it has been completely processed. However, some commands do not hold off the processing of subsequent commands. These commands are called overlapped commands. At times, overlapped commands require synchronization. The Attribute Summary for each command indicates whether it requires synchronization.

The module uses the No Pending Operation (NPO) flag to keep track of overlapped commands that are still pending (that is, not completed). The NPO flag is reset to 0 when an overlapped command is pending. It is set to 1 when no overlapped commands are pending. You cannot read the NPO flag directly, but you can use *OPC and *OPC? to tell when the flag is set to 1.

If you use *OPC, bit 0 of the Standard Event Status event register is set to 1 when the NPO flag is set to 1. This allows the instrument to generate a service request when all pending overlapped commands are completed (assuming you have enabled bit 0 of the Standard Event Status register and bit 5 of the Status Byte register).

If you use *OPC?, +1 is placed in the output queue when the NPO flag is set to 1. This allows you to effectively pause the controller until all pending overlapped commands are completed. It must wait until the response is placed in the queue before it can continue.

Note The *CLS and *RST commands cancel any preceding *OPC command or query. Pending overlapped commands are still completed, but you can no longer determine when.

*RST command

Executes a device reset.

Command Syntax: *RST

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This command returns the instrument to a reset state. In addition, *RST cancels any pending *OPC command or query.

The reset state is the same as the preset state. The preset state of each command is listed in the Attribute Summary.

The following are *not* affected by this command:

- The error queue
- The state of all enable registers
- The state of all transition registers

*SRE command/query

Sets or queries bits in the Service Request enable register.

Command Syntax: *SRE <Mask>

<Mask>::=number, limits: 0:255

Query Syntax: *SRE?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This command allows you to set bits in the Service Request enable register. Assign a decimal weight to each bit you want set (to 1) according to the following formula:

$$2^{(\text{bit\_number})}$$

with acceptable values for bit_number being 0 through 7. Add the weights and then send the sum with this command.

Note The module ignores the setting you specify for bit 6 of the Service Request enable register. This is because the corresponding bit of the Status Byte register is always enabled.

The module requests service from the active controller when one of the following occurs:

- A bit in the Status Byte register changes from 0 to 1 while the corresponding bit of the Service Request enable register is set to 1.
- A bit in the Service Request enable register changes from 0 to 1 while the corresponding bit of the Status Byte register is set to 1.

The query returns the current state of the Service Request enable register. The state is returned as a sum of the decimal weights of all set bits.

*STB? query

Reads the Status Byte register.

Query Syntax: *STB?

Return Format: Integer

Attribute Summary: Preset State: variable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This command allows you to set bits in the Status Byte register. The state is returned as a sum of the decimal weights of all set bits. The decimal weight for each bit is assigned according to the following formula:

$$2^{(\text{bit\_number})}$$

with acceptable values for bit_number being 0 through 7.

The register is not cleared by this query. To clear the Status Byte register, you must send the *CLS command.

For more information on the Status Byte register, see “The HP E2749’s Register Sets” in the previous chapter.

*TST? query

Performs a selftest on the instrument hardware and returns the results.

Query Syntax: *TST?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This command performs tests on the internal main board by sending the command DIAGnostic:BOARd:MAIN?

If the return is 0, the A1 assembly is working properly.

If the return is not 0, the A1 assembly is probably faulty.

See the DIAGnostic commands for additional diagnostic tests.

*WAI command

Holds off processing of subsequent commands until all preceding commands have been processed.

Command Syntax: *WAI

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: Use *WAI to hold off the processing of subsequent commands until all pending overlapped commands have been completed.

Some commands are processed sequentially by the instrument. A sequential command holds off the processing of any subsequent commands until it has been completely processed. However, some commands do not hold off the processing of subsequent commands; they are referred to as overlapped commands. *WAI ensures that overlapped commands are completely processed before subsequent commands (those sent after *WAI) are processed.

DIAGnostic:FIBRe:LOGin? query

Returns a list of logged in node information.

Query Syntax: DIAGnostic:FIBRe:LOGin?

Return Format: String

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: This command returns a list of all nodes currently logged in to the Fibre Channel fabric. The first column contains the address identifier. Columns two and three contain the World Wide Name for the port.

DIAGnostic:FIBRe:LOOPback command

Performs a loopback test.

Command Syntax: DIAGnostic:FIBRe:LOOPback <block size>,<count>

<block size> ::= number, limits 4 to 4096, must be multiple of 4

<count> ::= number, limits 1 to 2147483648

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: The number of blocks specified by the <count> parameter are sent to this HP E2749A module using the Systran FibreXpress Lightweight Protocol. The size in bytes of each block is specified by the <block size> parameter.

The DIAG:FIBR:LOOP:MODe command can be used to choose the loopback data path.

After the loopback test is started using the DIAG:FIBR:LOOP command, the DIAG:FIBR:LOOP:COUN? query can be used monitor the progress of the loopback test. If the count value doesn't increase the loopback test has failed. The final response of the DIAG:FIBR:LOOP:COUN? query should match the <count> parameter of the DIAG:FIBR:LOOP command.

Note

This command should not be used to measure the data transmission or reception performance due to the high amount of internal software overhead involved in implementing the test.

If the test fails, you may have to perform an 'iclear vxi' to re-establish the module's normal operation.

To abort the test before normal completion, perform an 'iclear vxi.'

DIAGnostic:FIBRe:LOOPback:COUNt? query

Returns the current loopback test block count.

Query Syntax: DIAGnostic:FIBRe:LOOPback:COUNt?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: This query will return the correct block count for a loopback test previously started using the “DIAG:FIBR:LOOP: command. As the loopback test runs, the count value returned by this query should have larger values with each successive query until the test completes.

DIAGnostic:FIBRe:LOOPback:MODe command/query

Selects the loopback data path.

Command Syntax: DIAGnostic:FIBRe:LOOPback:MODe <mode>

<mode> ::= OFF | TACHyon | GLM

Query Syntax: DIAGnostic:FIBRe:LOOPback:MODe?

Return Format: Char

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: Sending "DIAG:FIBR:LOOP:MOD TACH" reinitializes the Fibre Channel hardware and puts the HP Tachyon Fibre Channel ASIC in internal loopback mode.

Sending "DIAG:FIBR:LOOP:MOD GLM" reinitializes the fibre channel hardware and puts the HP Tachyon Fibre Channel ASIC in external loopback mode which informs the GLM to establish a loopback connection.

Sending "DIAG:FIBR:LOOP:MOD OFF" reinitializes the fibre channel hardware and removes any loopback connections.

Caution The loopback mode should only be changed for installation testing. Changing the loopback mode will cause the HP Tachyon Fibre Channel ASIC to take the loop down and then restore its operation with the new mode. Selecting the mode "GLM" or "TACHyon" may cause the rest of Fibre Channel nodes in a network to operate incorrectly..

DIAGnostic:FIBRe:LWSYstran:VERSion? query

Returns the revision number of the Systran driver.

Query Syntax: DIAGnostic:FIBRe:LWSYstran:VERSion?

Return Format: String

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: Returns the revision number of the firmware embedded Systran Corporation Lightweight Protocol driver.

DIAGnostic:LBUS:CONSume? query

Tests the a local bus data transfer to the module.

Query Syntax: DIAGnostic:LBUS:CONSume? <Logical Address>

<Logical Address> ::= number
limits 0-255

Example Queries: DIAGNOSTIC:LBUS:CONSUME? 32
diag:lbus:cons? 96

Return Format: String

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description:. This test requires more than one HP E2749 module. This command is sent to the HP E2749 on the right of two adjacent HP E2749s and tests the ability to transfer data from the local bus to the module.

<Logical Address> specifies the VXI logical address of the HP E2749 to the left of this module

Failures return a string describing the error.

DIAGnostic:LBUS:GENerate? query

Tests the local bus data transfer from the module.

Query Syntax: DIAGnostic:LBUS:GENerate? <Logical Address>

Example Queries: DIAGNOSTIC:BUS:GENERATE? 64
diag:bus:gen? 136

Return Format: String

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: This test requires more than one HP E2749 module to perform. This command is sent to the HP E2749 on the left of two adjacent HP E2749s and tests the ability to transfer data from the module to the local bus.

<Logical Address> specifies the VXI logical address of the HP E2749 to the left of this module

Failures return a string describing the error.

MMEMory:SESSion[1|2|...|12]:PROTocol command

This command selects the communication protocol to be used for the specified session.

Command Syntax: MMEMory:SESSion[1|2|...|12]:PROTocol<Protocol>,<receiver WWN hi>,<receiver WWN lo>,<application ID>

<Protocol>::=LWSYstran

<receiver WWN hi>::=number, limits 0:4294967296

<receiver WWN lo>::=number, limits 0:4294967296

<application ID>::=number, limits 0:65536

Query Syntax: MMEMory:SESSion[1|2|...|12]:PROTocol?

Example Command: MMEM:SESS1:PROT LWSYstran

Response Format: CHAR,Integer,Integer,Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: yes
SCPI Compliance: instrument-specific

Description: This command selects the communication protocol to be used. For the HP E2749A the only protocol supported is Systran's FibreXpress Lightweight protocol. If the session is going to be used to transmit data, the <receiver WWN hi> and <receiver WWN lo> parameters specify the Fibre Channel WWN of the receiving node. The <application ID> is required when transmitting or receiving.

SEQuence[1|2|3|4]:ADD command

Append an operation to the specified Sequence.

Command Syntax: SEQuence[1|2|3|4]:ADD <Operation>, <Count>, <Address>, <Misc>

<Operation>::=number
limits: 0:65535

<Count>::=number
limits: 0:4294967295

<Address>::=number
limits: 0:4294967295

<Misc>::=number
limits: 0:4294967295

Example Commands: SEQUENCE:ADD #h1000,#h10,0,#h03000800
seq3:add #h3012,65536,2048,0

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: Add the specified operation to the end of the specified Sequence. The maximum number of operations in a single Sequence is 100. The number of operations currently in the Sequence may be determined by sending the SEQ:SIZE? command.

The list of Sequence operations should be cleared by sending the SEQ:DEL:ALL command before adding new Sequence operations using this command.

Note For a list and detailed description of all Sequence operations which may be added using this command and an explanation of the above parameters see the chapter entitled “Using Sequence Operations with the HP E2749”.

SEQuence[1|2|3|4]:BEGin command

Begin a Sequence for transmitting or receiving data.

Command Syntax: SEQuence[1|2|3|4]:BEGin <Type>, <Byte Count>, <Session>

<Type>::=TRANsmit|RECeive

<Byte Count>::=number
limits: 1:9223372036854775807

<Session>::=number
limits: 1:12

Example Commands: SEQ:BEG TRAN,262144,1
sequence3:begin transmit,#h200000000,2

Attribute Summary: Preset State: not applicable
Synchronization Required: yes
SCPI Compliance: instrument-specific

Description: Begin execution of the specified Sequence (SEQ[1|2|3|4]). A Session must already be set up before sending this command. See the commands MMEM:SESS:*.

<Type> indicates whether the Sequence will be transmitting or receiving data. The internal software needs to be told this in order to get data flowing in the right direction. When <type> is TRANSMIT, data will be collected from modules in the VXI chassis using the VXIbus or the local bus. The collected data will be transmitted over Fibre Channel to a receiving application. When <type> is RECEIVE, data will be received over Fibre Channel from a transmitting application. The received data will be distributed to modules in the VXI chassis over the VXIbus or the local bus.

<Byte Count> is a 64-bit integer which indicates the total number of bytes which will be transferred by the Sequence. Once this byte count is reached, the Sequence will terminate.

<Session> indicates which Session will be used for this Sequence. The commands to initialize the Session must already have been successfully sent to the module. See MMEM:SESS:*.

SEQuence[1|2|3|4]:DELete:ALL command

Remove all elements from the specified Sequence list.

Command Syntax: SEQuence[1|2|3|4]:DELete:ALL

Example Commands: SEQUENCE4:DELETE:ALL
seq2:del:all

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: Delete all elements of the Sequence. This is the only command which removes elements from a Sequence. You should send this command before beginning to add elements to a Sequence.

SEQuence[1|2|3|4]:SIZE? query

Return the number of elements in the Sequence.

Query Syntax: SEQuence[1|2|3|4]:SIZE?

Example Queries: SEQ2:SIZE?
sequence:size?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: The number returned should equal the number of times SEQ:ADD has been sent since the last SEQ:DEL:ALL, *RST, or powerup. The maximum number of elements in a Sequence is 100.

SEQuence[1|2|3|4]:TRANsferred? query

Return the number of bytes transferred in the Sequence.

Query Syntax: SEQuence[1|2|3|4]:TRANsferred?

Example Queries: SEQUENCE2:TRANSFERRED?
seq4:tran?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: If the Sequence completed without errors and was not aborted, the returned value will be almost equal to the number of bytes specified with the SEQ:BEG command. Otherwise, the returned value will indicate the number of bytes which were successfully transferred.

Note that the sequence operation code 602X can be used within a sequence to write the current byte count to a memory buffer.

STATus:OPERation:CONDition? query

Reads the Operation Status condition register.

Query Syntax: STATus:OPERation:CONDition?

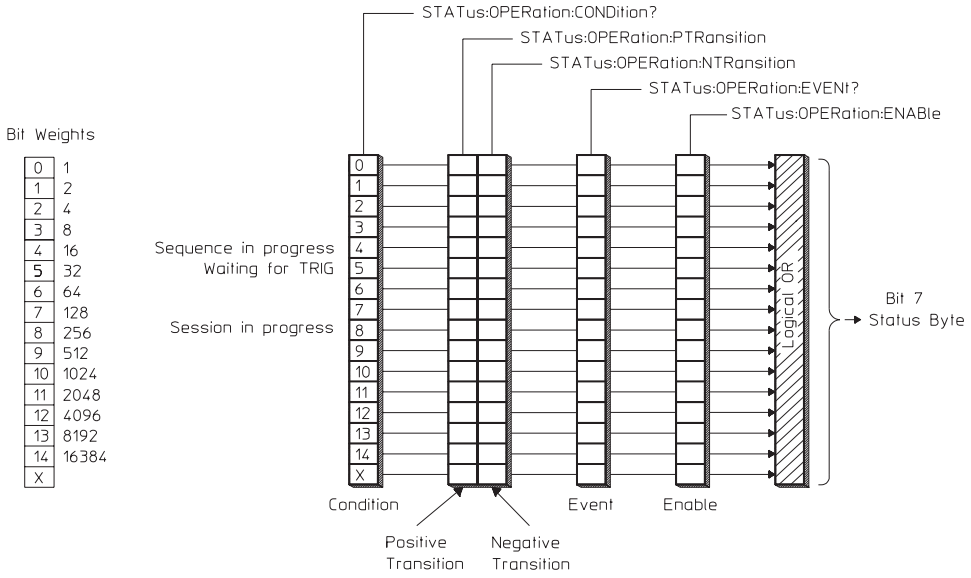
Example Queries: STATUS:OPERATION:CONDITION?
status:operation:condition?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This query returns the sum of the decimal weights of all bits currently set to 1 in the Operation Status condition register. (The decimal weight of a bit is 2^n, where n is the bit number.)

See “Operation Status Register Set” in the previous chapter for a definition of bits in the register set.



STATus:OPERation:ENABle command/query

Sets and queries bits in the Operation Status enable register.

Command Syntax: STATus:OPERation:ENABle <Bit Mask>

<Bit Mask>::=number
limits: 0:32767

Example Commands: STATUS:OPER:ENAB 304
status:operation:enable 32

Query Syntax: STATus:OPERation:ENABle?

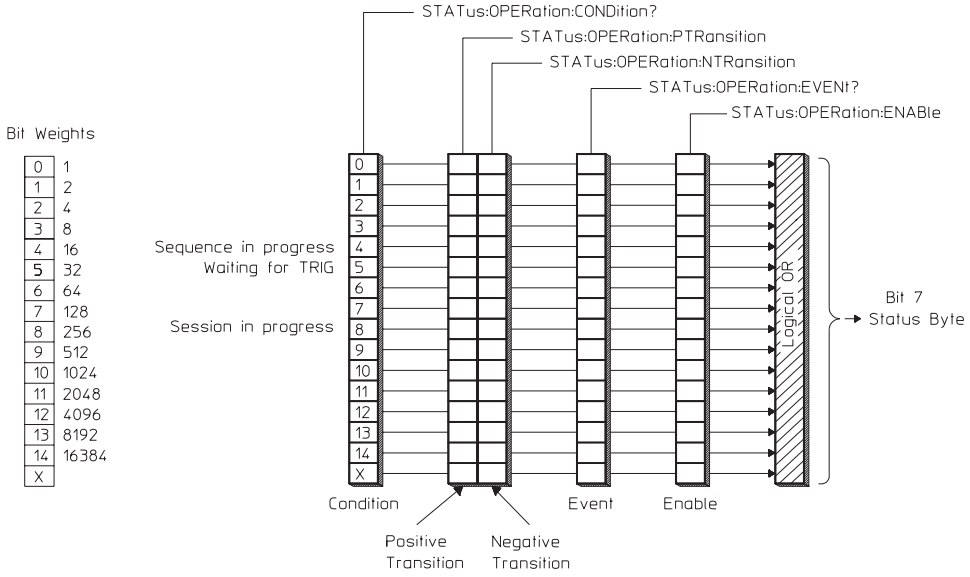
Return Format: Integer

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: To set a single bit in the Operation Status enable register to 1, send the bit's decimal weight with this command. To set more than one bit to 1, send the sum of the decimal weights of all the bits. (The decimal weight of a bit is 2^n, where n is the bit number.)

All bits are initialized to 0 on powerup or when the STAT:PRES command is sent. However, the current setting of bits is *not* modified when you send the *RST command.

See "Operation Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:OPERation[:EVENt]? query

Reads and clears the Operation Status event register.

Query Syntax: STATus:OPERation[:EVENt]?

Example Queries: STATUS:OPERATION?
stat:oper:even?

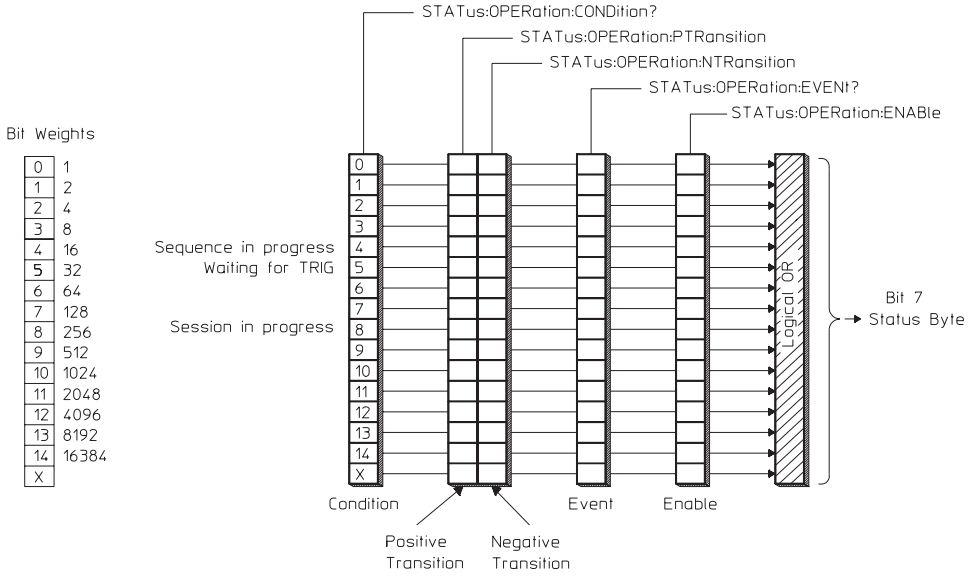
Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This query returns the sum of the decimal weights of all bits currently set to 1 in the Operation Status event register. (The decimal weight of a bit is 2^n, where n is the bit number.)

Note The Operation Status event register is automatically cleared after it is read by this query.

See "Operation Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:OPERation:NTRansition command/query

Sets and queries bits in the Operation Status negative transition register.

Command Syntax: STATus:OPERation:NTRansition <Bit mask>

<Bit mask>::=number
limits: 0:32767

Example Commands: STAT:OPER:NTR 256
status:operation:ntransition 48

Query Syntax: STATus:OPERation:NTRansition?

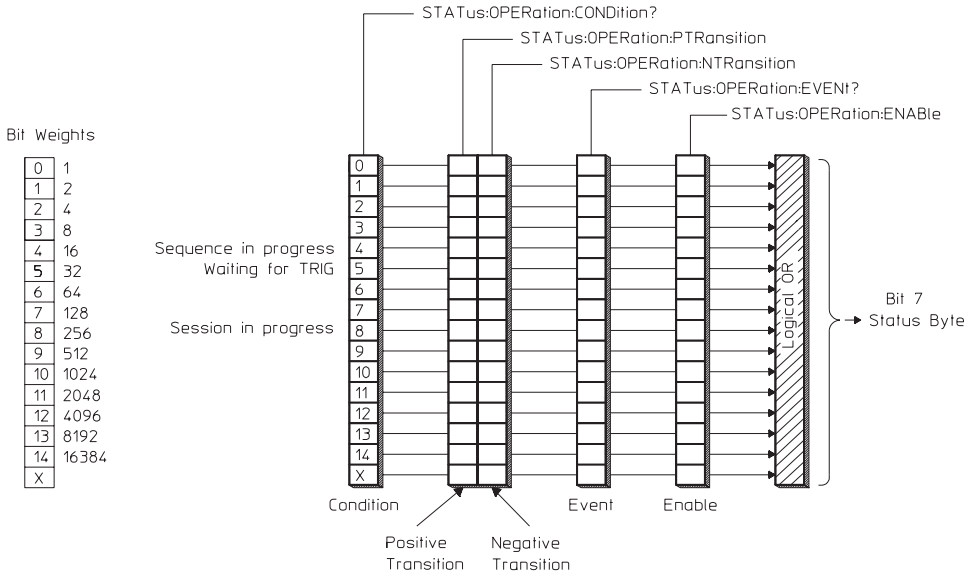
Return Format: Integer

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: To set a single bit in the Operation Status negative transition register to 1, send the bit's decimal weight with this command. To set more than one bit to 1, send the sum of the decimal weights of all the bits. (The decimal weight of a bit is 2^n, where n is the bit number.)

All bits are initialized to 0 on powerup or when the STAT:PRES command is sent. However, the current setting of bits is *not* modified when you send the *RST command.

See "Operation Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:OPERation:PTRansition command/query

Sets and queries bits in the Operation Status positive transition register.

Command Syntax: STATus:OPERation:PTRansition <Bit mask>

<Bit mask>::=number
limits: 0:32767

Example Commands: STAT:OPER:PTR 304
status:operation:ptransition 32

Query Syntax: STATus:OPERation:PTRansition?

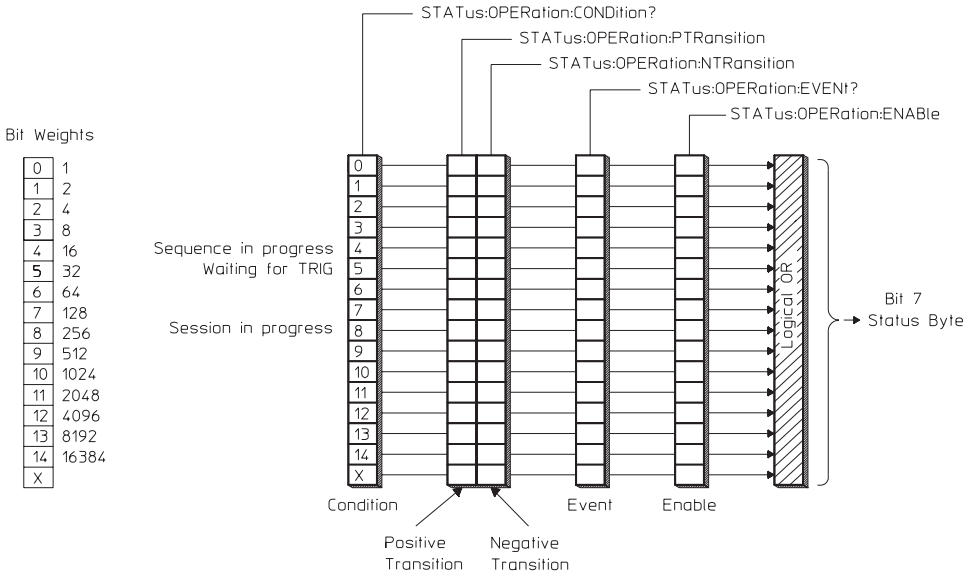
Return Format: Integer

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: To set a single bit in the Operation Status positive transition register to 1, send the bit's decimal weight with this command. To set more than one bit to 1, send the sum of the decimal weights of all the bits. (The decimal weight of a bit is 2^n, where n is the bit number.)

All bits are initialized to 1 on powerup or when the STAT:PRES command is sent. However, the current setting of bits is *not* modified when you send the *RST command.

See "Operation Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:PRESet command

Sets bits in most enable and transition registers to their default state.

Command Syntax: STATus:PRESet

Example Commands: STATUS:PRESET
stat:pres

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: STATUS:PRESet has the effect of bringing all events to the second level register sets (Questionable Status and Operation Status) *without* creating an SRQ or reflecting events in a serial poll.

It also affects these register sets (Questionable Status and Operation Status) as follows:

- Sets all enable register bits to 0.
- Sets all positive transition register bits to 1.
- Sets all negative transition register bits to 0.

STATus:QUEStionable:CONDition? query

Reads the Questionable Status condition register.

Query Syntax: STATus:QUEStionable:CONDition?

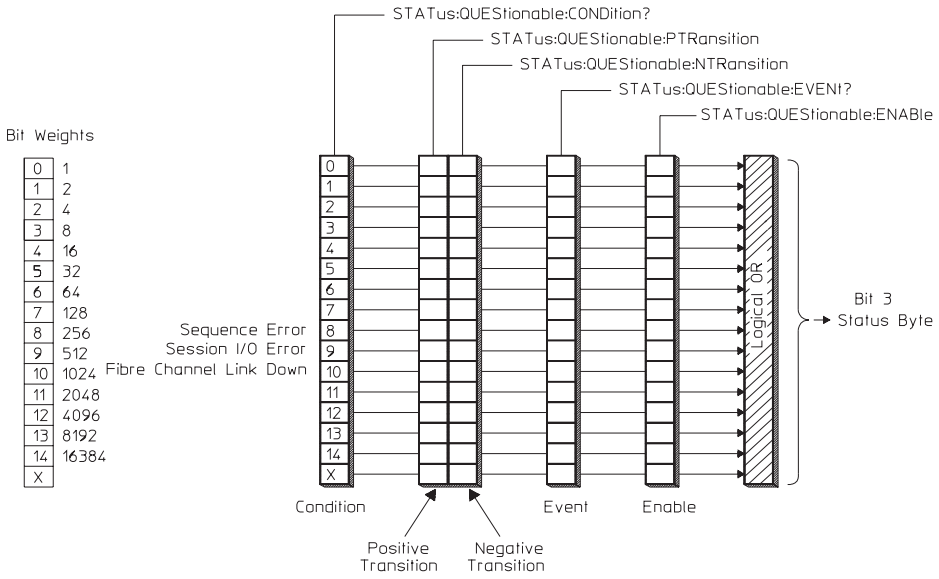
Example Queries: STAT:QUES:COND?
status:questionable:condition?

Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This query returns the sum of the decimal weights of all bits currently set to 1 in the Questionable Status condition register. (The decimal weight of a bit is 2^n, where n is the bit number.)

See "Questionable Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:QUEStionable:ENABle command/query

Sets and queries bits in the Questionable Status enable register.

Command Syntax: STATus:QUEStionable:ENABle <Bit Mask>

<Bit Mask>::=number
limits: 0:32767

Example Commands: STAT:QUES:ENAB 256
status:questionable:enable 512

Query Syntax: STATus:QUEStionable:ENABle?

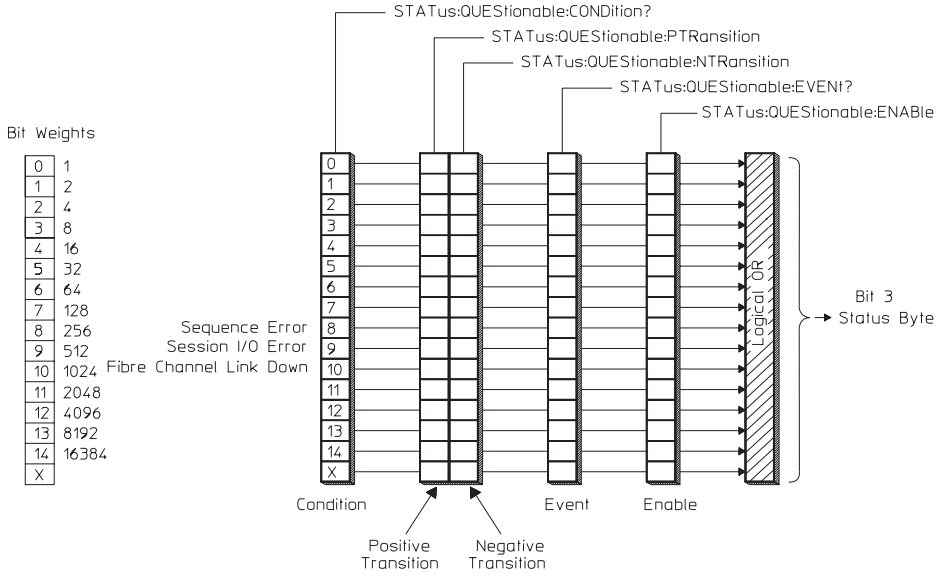
Return Format: Integer

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: To set a single bit in the Questionable Status enable register to 1, send the bit's decimal weight with this command. To set more than one bit to 1, send the sum of the decimal weights of all the bits. (The decimal weight of a bit is 2^n, where n is the bit number.)

All bits are initialized to 0 on powerup or when the STAT:PRES command is sent. However, the current setting of bits is *not* modified when you send the *RST command.

See "Questionable Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:QUEStionable[:EVENt]? query

Reads and clears the Questionable Status event register.

Query Syntax: STATus:QUEStionable[:EVENt]?

Example Queries: STATUS:QUESTIONABLE:EVENT?
status:questionable?

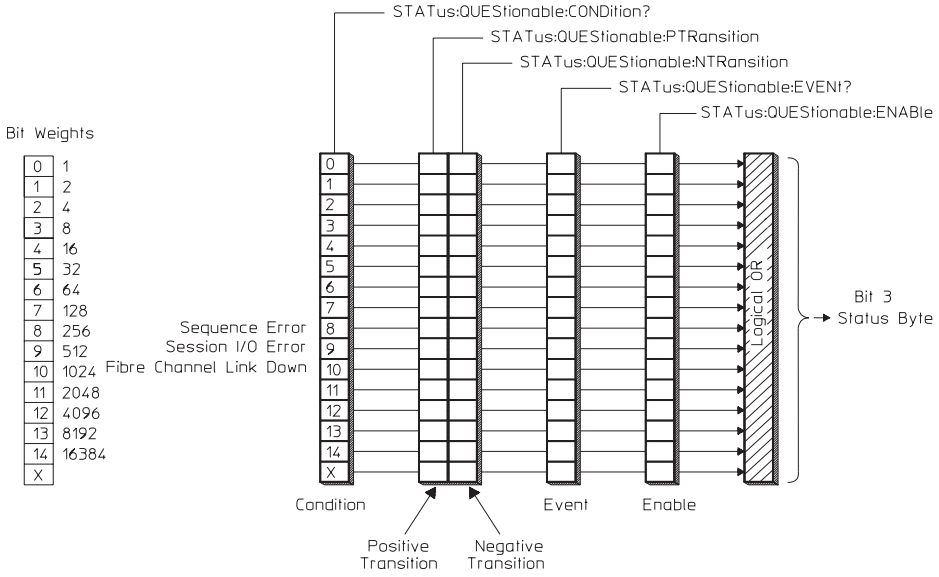
Return Format: Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: This query returns the sum of the decimal weights of all bits currently set to 1 in the Questionable Status event register. (The decimal weight of a bit is 2^n, where n is the bit number.)

Note The Questionable Status event register is automatically cleared after it is read by this query.

See "Questionable Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:QUEStionable:NTRansition command/query

Sets and queries bits in the Questionable Status negative transition register.

Command Syntax: STATus:QUEStionable:NTRansition <Bit mask>

<Bit mask>::=number
limits: 0:32767

Example Commands: STAT:QUES:NTR 768
Status:Questionable:Ntransition 256

Query Syntax: STATus:QUEStionable:NTRansition?

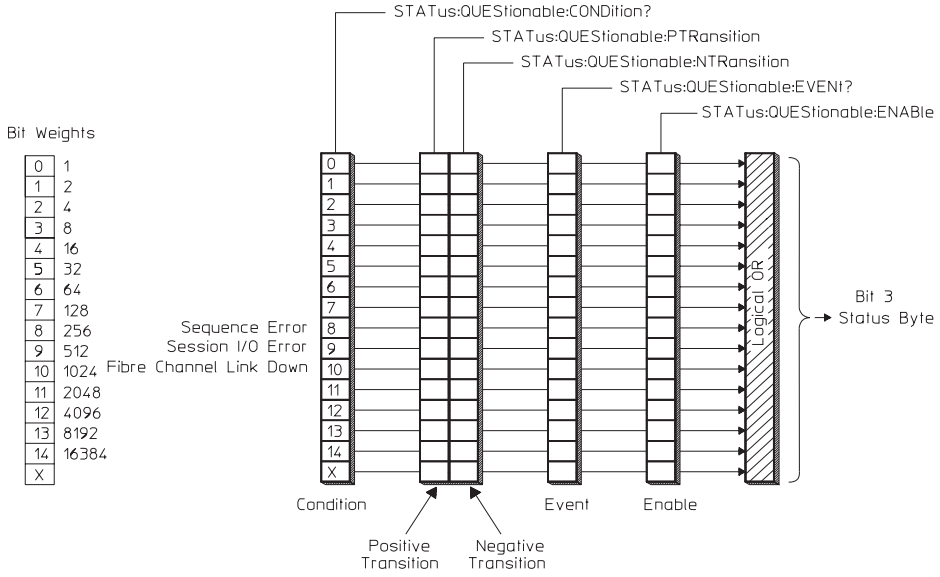
Return Format: Integer

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: To set a single bit in the Questionable Status negative transition register to 1, send the bit's decimal weight with this command. To set more than one bit to 1, send the sum of the decimal weights of all the bits. (The decimal weight of a bit is 2^n, where n is the bit number.)

All bits are initialized to 0 on powerup or when the STAT:PRES command is sent. However, the current setting of bits is *not* modified when you send the *RST command.

See "Questionable Status Register Set" in the previous chapter for a definition of bits in the register set.



STATus:QUEStionable:PTRansition command/query

Sets and queries bits in the Questionable Status positive transition register.

Command Syntax: STATus:QUEStionable:PTRansition <Bit mask>

<Bit mask>::=number
limits: 0:32767

Example Commands: STATUS:QUESTIONABLE:PTRANSITION 256
stat:ques:ptr 512

Query Syntax: STATus:QUEStionable:PTRansition?

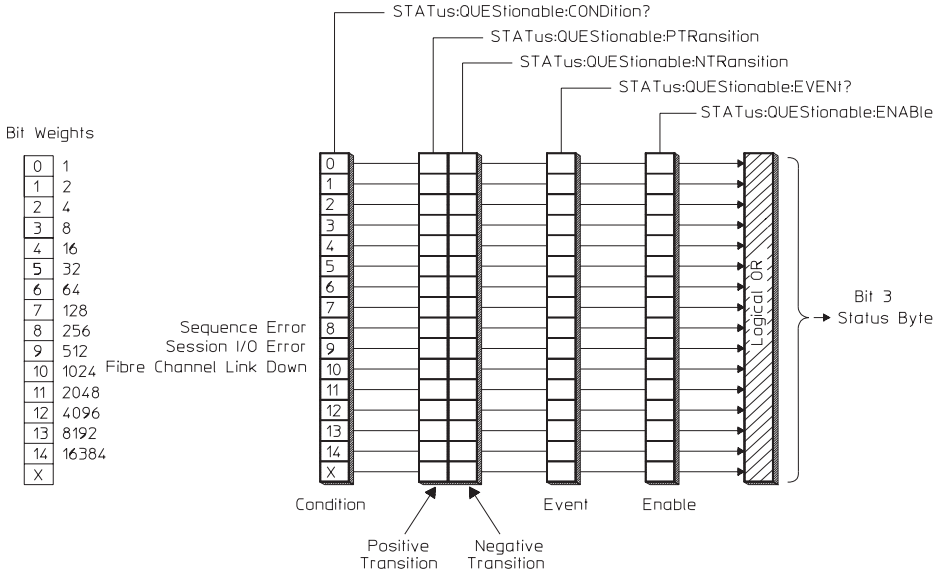
Return Format: Integer

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: To set a single bit in the Questionable Status positive transition register to 1, send the bit's decimal weight with this command. To set more than one bit to 1, send the sum of the decimal weights of all the bits. (The decimal weight of a bit is 2^n, where n is the bit number.)

All bits are initialized to 1 on powerup or when the STAT:PRES command is sent. However, the current setting of bits is *not* modified when you send the *RST command.

See "Questionable Status Register Set" in the previous chapter for a definition of bits in the register set.



SYSTem:ABORt command

Aborts a data transfer Session and/or Sequence.

Command Syntax: SYSTem:ABORt

Example Commands: SYSTEM:ABORT
syst:abor

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: Any data transfer Session in progress is aborted. The Session data structures will not be altered. The local bus is placed into the reset state. Any Sequence in progress is aborted. Sequence data structures are updated such that a SEQ:TRAN? query will correctly indicate the number of bytes actually transferred during the Sequence.

SYSTem:COMMunicate:WWN[:SELF]: ADDRess command/query

Stores the WWN for this HP E2749 module into ROM.

Command Syntax: SYSTem:COMMunicate:WWN[:SELF]:ADDRess <WWN hi>,<WWN lo>

<WWN hi>::=number, limits 0:4294967296

<WWN lo>::=number, limits 0:4294967296

Example Command: SYSTEM:COMMUNICATE:WWN:ADDRESS #h22222222,#h00001001

Query Syntax: SYSTem:COMMunicate:WWN:ADDRess?

Return Format: Integer,Integer

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: The command sets or queries the Fibre Channel World Wide Name (WWN) of this node. The WWN is a 64-bit value (specified in the command as the upper 32 bits and the lower 32 bits) that uniquely identifies this node in the Fibre Channel domain.

Note The ROM programming enable switch must be in the service position (1) to set the WWN.

Caution Setting the module's WWN involves reprogramming one of the flash ROMs on the module. This takes about 45 seconds. If the VXI chassis is turned off or reset (using the 'iclear vxi' command) during this operation, the flash ROM may be incorrectly programmed and render the module useless until the ROM is replaced.

Once set to a unique value compatible with your Fibre Channel network configuration, the WWN should not have to be changed.

SYSTem:ERRor? query

Returns one error message from the module's error queue.

Query Syntax: SYSTem:ERRor?

Example Queries: SYSTEM:ERROR?
syst:err?

Return Format: Integer
STRING

Attribute Summary: Preset State: not affected by Preset
Synchronization Required: no
SCPI Compliance: confirmed

Description: The error queue temporarily stores up to 10 error messages. When you send the SYST:ERR query, one message is moved from the error queue to the output queue so your controller can read the message. The error queue delivers messages to the output queue in the order received.

If more than 10 error messages are reported before any are read from the queue, the oldest error messages are saved. The last error message indicates that too many error messages were received for the queue.

Note The error queue is cleared when you turn on or reset the VXI system and when you send the *CLS command.

SYSTem:VERSion? query

Returns the SCPI version to which the module complies.

Query Syntax: SYSTem:VERSion?

Example Queries: SYSTEM:VERSION?
syst:vers?

Return Format: YYYY.V

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: confirmed

Description: The Ys represent the SCPI year-version and the V represents the revision number for that year.

The HP E2749 will return 1994.0.

VINStrument[:CONFigure]:LBUS [:MODE] RESet|NORMal|PIPE — command/query

Configures the local bus.

Command Syntax: VINStrument[:CONFigure]:LBUS[:MODE] <Lbus Mode>

<Lbus Mode>::=RESet|PIPE|NORMal

Example Commands: VINSTRUMENT:CONFIGURE:LBUS:MODE RESET
vins:lbus norm

Query Syntax: VINStrument[:CONFigure]:LBUS[:MODE]?

Return Format: CHAR

Attribute Summary: Preset State: RESet
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: The local bus interface has strict requirements as to the order in which modules in a VXI mainframe have their local bus interface reset. On powerup, or whenever any module in the mainframe is put in the reset state, all modules should be placed into the reset state from left to right. Then all modules can be put into the un-reset state from left to right.

Sending VINStrument:CONfigure:LBUS:MODE RESet, places the HP E2749 local bus interface into the LBUS RESet state. Sending either PIPE or NORMAL takes the local bus interface out of the LBUS RESet state. In order for the HP E2749 to use the local bus, the mode must be set to NORMal.

Sending VINStrument:LBUS PIPE puts the local bus interface into a state such that all local bus data from the module to the left is automatically routed to the module on the right. This is useful if you want to route local bus data past the HP E2749 to other modules rather than have the HP E2749 participate in any local bus throughput or playback. In this mode, the local bus cannot be used in a Sequence operation, or via the LBUS:READ:BUFFer or LBUS:WRITe:BUFFer commands. The behavior is undefined if this value is set to PIPE then a Sequence operation is executed which either reads from or writes to the local bus.

When transitioning from PIPE to NORMal mode one additional block will be piped after the change. In order to make this transition easier to coordinate it is best to reset the local bus on all modules then go to NORMal.

This command may be used instead of VINStrument:LBUS:RESet to place the HP E2749 local bus in the un-reset state.

VINStrument:LBUS:RESet command

Resets the local bus.

Command Syntax: VINStrument:LBUS:RESet

Example Commands:
```
VINSTRUMENT:LBUS:RESET
OUTPUT 70918;"vins:lbus:res
```

Attribute Summary: Preset State: not applicable
Synchronization Required: no
SCPI Compliance: instrument-specific

Description: The local bus interface has strict requirements as to the order in which modules in a VXI mainframe have their local bus interface reset. On powerup or whenever any module in the mainframe is put in the reset state, all modules should be placed into the reset state from left to right. Then all modules can be put into the un-reset state from left to right.

This command toggles the local bus reset state for the HP E2749; first going into the reset state, then back out. Once this is completed the local bus mode is NORMal.

This command is *not* required if you use the VINStrument:CONFigure:LBUS:MODE command to configure the local bus.

Errors

SCPI Command Errors

Error Number	Description
-100	Command error. This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that a Command Error as defined in *IEEE 488.2*, 11.5.1.1.4 has occurred.
-101	Invalid character. A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of errors -114, -121, -141, and perhaps others.
-102	Syntax error. An unrecognized command or data type was encountered; for example, a string was received when the device does not accept strings.
-103	Invalid separator. The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit, *EMC 1 :CH1:VOLTS 5.
-104	Data type error. The parser recognized a data element different than one allowed; for example, numeric or string data was expected but block data was encountered.
-105	GET not allowed. A Group Execute Trigger was received within a program message (see *IEEE 488.2*, 7.7).
-108	Parameter not allowed. More parameters were received than expected for the header; for example, the *EMC common command only accepts one parameter, so receiving *EMC 0,,1 is not allowed.
-109	Missing parameter. Fewer parameters were received than required for the header; for example, the *EMC common command requires one parameter, so receiving *EMC is not allowed.
-110	Command header error. An error was detected in the header. This error message is used when the device cannot detect the more specific errors described for errors -111 through -119.
-111	Header separator error. A character which is not a legal header separator was encountered while parsing the header; for example, no white space followed the header, thus *GMC"MACRO" is an error.
-112	Program mnemonic too long. The header contains more than twelve characters (see *IEE488.2*, 7.6.1.4.1).
-113	Undefined header. The header is syntactically correct, but it is undefined for this specific device; for example, *XYZ is not defined for any device.
-114	Header suffix out of range. The value of a numeric suffix attached to a program mnemonic, see Syntax and Style section 6.2.5.2, makes the header invalid.
-120	Numeric data error. This error, as well as errors -121 through -129, are generated when parsing a data element which appears to be numeric, including the nondecimal numeric types. This particular error message is used if the device cannot detect a more specific error.
-121	Invalid character in number. An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric or a "9" in octal data.
-123	Exponent too large. The magnitude of the exponent was larger than 32000 (see *IEEE 488.2*, 7.7.2.4.1).
-124	Too many digits. The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros (see *IEEE 488.2*, 7.7.2.4.1).

SCPI Command Errors

-128 Numeric data not allowed. A legal numeric data element was received, but the device does not accept one in this position for the header.

-130 Suffix error. This error, as well as errors -131 through -139, are generated when parsing a suffix. This particular error message is used if the device cannot detect a more specific error.

-131 Invalid suffix. The suffix does not follow the syntax described in *IEEE 488.2*, 7.7.3.2, or the suffix is inappropriate for this device.

-134 Suffix too long. The suffix contained more than 12 characters (see *IEEE 488.2*, 7.7.3.4).

-138 Suffix not allowed. A suffix was encountered after a numeric element which does not allow suffixes.

-140 Character data error. This error, as well as errors -141 through -149, are generated when parsing a character data element. This particular error message is used if the device cannot detect a more specific error.

-141 Invalid character data. Either the character data element contains an invalid character or the particular element received is not valid for the header.

-144 Character data too long. The character data element contains more than twelve characters (see *IEEE 488.2*, 7.7.1.4).

-148 Character data not allowed. A legal character data element was encountered where prohibited by the device.

-150 String data error. This error, as well as errors -151 through -159, are generated when parsing a string data element. This particular error message is used if the device cannot detect a more specific error.

-151 Invalid string data. A string data element was expected, but was invalid for some reason (see *IEEE 488.2*, 7.7.5.2); for example, an END message was received before the terminal quote character.

-158 String data not allowed. A string data element was encountered but was not allowed by the device at this point in parsing.

-160 Block data error. This error, as well as errors -161 through -169, are generated when parsing a block data element. This particular error message is used if the device cannot detect a more specific error.

-161 Invalid block data. A block data element was expected, but was invalid for some reason (see *IEEE 488.2*, 7.7.6.2); for example, an END message was received before the length was satisfied.

-168 Block data not allowed. A legal block data element was encountered but was not allowed by the device at this point in parsing.

-170 Expression error. This error, as well as errors -171 through -179, are generated when parsing an expression data element. This particular error message is used if the device cannot detect a more specific error.

-171 Invalid expression. The expression data element was invalid (see *IEEE 488.2*, 7.7.7.2); for example, unmatched parentheses or an illegal character.

-178 Expression data not allowed. A legal expression data was encountered but was not allowed by the device at this point in parsing.

-181 Invalid outside macro definition. Indicates that a macro parameter placeholder ($<$number) was encountered outside of a macro definition.

-183 Invalid inside macro definition. Indicates that the program message unit sequence, sent with a *DDT or *DMC command, is syntactically invalid (see *IEEE 488.2*, 10.7.6.3).

SCPI Execution Errors

Error Number	Description
-200	Execution error. This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that an Execution Error as defined in *IEEE 488.2*, 11.5.1.1.5 has occurred.
-220	Parameter error. Indicates that a program data element related error occurred. This error message is used when the device cannot detect the more specific errors described for errors -221 through -229.
-221	Settings conflict. Indicates that a legal program data element was parsed but could not be executed due to the current device state (see *IEEE 488.2*, 6.4.5.3 and 11.5.1.1.5.)
-222	Data out of range. Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range as defined by the device (see *IEEE 488.2*, 11.5.1.1.5.)
-223	Too much data. Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.
-224	Illegal parameter value. Used where exact value, from a list of possibilities, was expected.
-240	Hardware error. Indicates that a legal program command or query could not be executed because of a hardware problem in the device. Definition of what constitutes a hardware problem is completely device-specific. This error message is used when the device cannot detect the more specific errors described for errors -241 through -249.
-272	Macro execution error. Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition (see *IEEE 488.2*, 10.7.6.3.)
-273	Illegal macro label. Indicates that the macro label defined in the *DMC command was a legal string syntax, but could not be accepted by the device (see *IEEE 488.2*, 10.7.3 and 10.7.6.2); for example, the label was too long, the same as a common command header, or contained invalid header syntax.
-276	Macro recursion error. Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive (see *IEEE 488.2*, 10.7.6.6).
-277	Macro redefinition not allowed. Indicates that a syntactically legal macro label in the *DMC command could not be executed because the macro label was already defined (see *IEEE 488.2*, 10.7.6.4).
-278	Macro header not found. Indicates that a syntactically legal macro label in the *GMC? query could not be executed because the header was not previously defined.

SCPI Device-Specific Errors

Error Number	Description
-310	System error. Indicates that some error termed “system error” by the device, has occurred. This code is device-dependent.
-311	Memory error. Indicates that an error was detected in the device’s memory. The scope of this error is device-dependent.
-315	Configuration memory lost. Indicates that nonvolatile configuration data saved by the device has be lost. The meaning of this error is device-specific.
-321	Out of memory.
-330	Self-test failed.

SCPI Device-Specific Errors

-350	Queue overflow. A specific code entered into the queue in lieu of the code that caused the error. This code indicates that there is no room in the queue and an error occurred but was not recorded.

SCPI Query Errors

Error Number	Description
-400	Query error. This is the generic query error for devices that cannot detect more specific errors. This code indicates only that a Query Error as defined in *IEEE 488.2*, 11.5.1.1.7 and 6.3 has occurred.
-410	Query INTERRUPTED. Indicates that a condition causing an INTERRUPTED Query error occurred (see *IEE 448.2*, 6.3.2.3); for example, a query followed by DAB or GET before a response was completely sent.
-420	Query UNTERMINATED. Indicates that a condition causing an UNTERMINATED Query error occurred (see *IEEE 488.2*, 6.3.2.2); for example, the device was addressed to talk and an incomplete program message was received.
-430	Query DEADLOCKED. Indicates that a condition causing an DEADLOCKED Query error occurred (see *IEEE 488.2*, 6.3.1.7); for example, both input buffer and output buffer are full and the device cannot continue.
-440	Query UNTERMINATED after indefinite response. Indicates that a query was received in the same program message after a query requesting an indefinite response was executed (see *IEEE 488.2*, 6.5.7.5).

HP E2749-Specific Errors

Error Number	Description
6207	Session busy. A SEQ:BEG command attempted to use a Session which was already performing some operation. Use of the Session Busy bit in the Operation Status register may help avoid this error.
6209	Sequence full. A SEQ:ADD was attempted on a Sequence which already contained the maximum number of Sequence operations.
6210	Sequence busy. A SEQ:BEG command was attempted when a Sequence was already running.
6211	Sequence empty. A SEQ:BEG command was attempted on a Sequence containing no Sequence operations.
6212	Local bus busy. The local bus chip was not in the paused state when a SEQ:BEG was attempted.
6218	Loopback test already running. A previous DIAG:FIBR:LOOP command has not completed.
6219	FXLP write error. The internal FXLP driver detected an error when writing to another node. The link is down, or the receiving application is not running or ready to receive data.

10

Using the HP E2749A with Systran FibreXpress™ Products

Getting Started

The HP E2749A Fibre Channel Interface module supports Systran Corporation's FibreXpress Lightweight Protocol (FXLP). Since the HP E2749 is operated using SCPI commands and not programmed by the user in the 'C' programming language, the module's firmware makes calls to the FXLP API functions when specific SCPI commands are executed.

Detailed information on FXLP is in the *FibreXpress FCLP Driver Programmer's Reference Guide,* available from Systran Corporation.

Compatible hardware and software products are available from Systran Corporation, 800-252-5601. Systran Corporation also has a Web site at www.systran.com

As of April 1, 1997, Systran FibreXpress products are available on the systems listed in the following table.

Processor	Operating System	Bus Interface
i960	VxWorks 5.3	PMC
MIPS	VxWorks 5.2	PMC
PowerPC	VxWorks 5.2	PMC
Pentium PC	Windows NT	PCI
Sun Sparc	Solaris 2.4/2.5	SBus

Firmware revision of Systran FXLP driver

The HP E2749A contains (in ROM) a version of the Systran FXLP driver. You can read the version number using the SCPI query
`DIAG:FIBR:LWSY:VERS?`

See the chapter titled "SCPI Command Reference" for details on this command.

SCPI commands required for HP E2749A data transmission

A minimum set of SCPI sommands must be sent to the HP E2749 to begin data transmission to a receiving node.

`MMEM:SESS 1:PROT LWSY,<WWW hi>,<WWW lo>,<receiver APID>`
where `<WWN hi>` and `<WWN lo>` identify the WWN of the receiving node, and `<receiver APID>` is the APID value used in the receiving application call to the `lp_create_ap()` function.

`SEQ1:ADD <sequence operation code>,<count>,<address>,<misc>`
where `<sequence operation code>` identifies the source (local VXI bus, VXI/VME bus, . . .) of the data.

`SEQ1:BEG TRAN,<byte count>,1`
where `<byte count>` is the total number of bytes to transmit.

SCPI commands required for HP E2749A data reception

A minimum set of SCPI sommands must be sent to the HP E2749 to begin receiving data transmitted from the transmitting node.

`MMEM:SESS 1:PROT LWSY, 0 0, <receiver APID>`
where `<receiver APID>` is the APID number used in the HP E2749 firmware's call to `lp_create_ap()` (and used in the transmitting application call to `lp_setup_chan()`).

`SEQ1:ADD <sequence operation code>,<count>,<address>,<misc>`
where `<sequence operation code>` identifies the destination (local VXI bus, VXI/VME bus, . . .) of the received data.

`SEQ1:BEG REC,<byte count>,1`
where `<byte count>` is the total number of bytes to receive.

Transmitting and receiving data

The next section includes two simple examples showing data transmission from the HP E2749A to a computer and data transmission from a computer to an HP E2749A. When the `SEQuence:BEGin` SCPI command is sent to the HP E2749 to start data transmission (or reception), the HP E2749 firmware makes calls to the firmware embedded FXLP API functions. These function calls are shown in the tables.

Transmitting Data From the HP E2749A

When you use the HP E2749 to transmit data to a computer the following sequence of operations must occur. The numbers in the first column refer to the notes following the table.

Note	TRANSMIT DATA APPLICATION SCPI Commands Sent to HP E2749 $ firmware function calls	RECEIVE DATA APPLICATION (computer) WWN: 0x2222222200000001 FXLP API calls on receiving computer
1		apid = 2;
2		apnum = lp_create_ap (fd, apid);
3		chnum = lp_wait_for_chan (fd, apnum, &transmitter_addr, N &max_blk_size);
4	MMEM:SESS1:PROT LWSYSTRAN, #h22222222,#h00000001,2	
5	SEQ1:DEL:ALL	
5	SEQ1:ADD #h3100,2048,0,0	
6	SEQ1:BEG TRANSMIT,65536,1	

7	$ apid = 1;
	$ receiver_addr.a_wwn.hi =
	0x22222222;
	$ receiver_addr.a_wwn.lo =
	0x00000001;
	$ receiver_addr.a_apid = 2;	

8	$ apnum = lp_create_ap(fd, apid);	
9	$ chnum = lp_setup_chan(fd, apnum, &receiver_addr, &max_blk_size);	
10		stat = lp_recv (fd, apnum, &buf, 2048, apnum, chnum, &numrec);
11	$ stat = lp_send (fd, &buf, 2048, apnum, chnum,&numsend);	
	$ stat = lp_send (....	stat = lp_recv (....
12	$ stat = lp_close_chan (fd, apnum,chnum);	
13		stat = lp_close_chan (fd, apnum, chnum);
	$ stat = lp_destroy_ap(fd, apnum);	stat = lp_destroy_ap(fd, apnum);
14		

Notes

1 The computer has chosen to receive data using application id (APID) with a value of 2.

2 The computer informs the FXLP driver that it will receive any data for APID 2. For the FXLP protocol to operate properly, the data receiving application should be started before the transmitting application.

3 The computer waits for a transmitter (e.g. an HP E2749A) to establish a connection. This call is blocking, that is, you won't return from this function until a transmitter makes a connection (by calling function `lp_setup_chan)`. The address of the transmitter that connected is returned in variable transmitter_addr.

4 The SCPI command `MMEM:SESS1:PROT` is sent to the HP E2749 to select the Systran Lightweight Protocol (LWSYSTRAN), and identify the WWN and APID, of the receiver. In this example we're assuming that the computer's WWN is `0x2222222200000001` and the receiver is using an APID value of 2.

5 Load a sequence into the HP E2749A. For this example the sequence operation `#h3100` is used to transmit dummy bytes. `2048` is the number to transmit each time the sequence operation is executed. In an actual application, sequence operations that transmit bytes from the Local Bus are more likely to be used.

6 Start transmitting 65536 bytes of data using Sequence 1 and Session 1. This SCPI command results in internal firmware calls to the FXLP API functions as described in the steps that follow.

7 This step sets variables that specify which receiving APID at which WWN we are to connect to. The parameters from the `MMEM:SESS:PROT` command are used in these internal function calls.

8 Set up an APID for the transmitter.

9 Connect to the receiving APID at the specified WWN — in this example WWN `2222222200000001` and APID (at the computer) `2`.

10 The HP E2749 firmware's internal call to `lp_setup_chan()` causes the computer's call to `lp_wait_for_chan()` to unblock. At this time the computer knows that a transmitting node has established a connection. The variable `transmitter_addr` will contain the WWN of the HP E2749A that established the connection. The computer calls `lp_rec()` to receive data.

11 The HP E2749 firmware calls `lp_send()` multiple times to transfer the 65536 bytes, as specified in the `SEQ:BEG` command, to the computer.

12 The HP E2749 firmware calls `lp_send()` multiple times to transfer the 65536 bytes, as specified in the `SEQ:BEG` command, to the computer.

13 The HP E2749 firmware calls `lp_close_chan()`.

14 The `lp_recv()` call on the computer side detects the closed channel and quits calling `lp_recv(). lp_close_chan()` is called to close the channel on the receiving end.

15 At this point 65536 bytes have been transferred from the HP E2749A to the computer.

Receiving Data From a Computer With The HP E2749A

When the HP E2749A is being used to receive data from a computer the following sequence of operations must occur.

Note	TRANSMIT DATA APPLICATION FXLP API calls on transmitting computer	RECEIVE DATA APPLICATION (HP E2749A) WWN: 0x2222222200000002 SCPI Commands Sent to HP E2749 $ firmware function calls
1		MMEM:SESS1:PROT LWSYSTRAN,0,0,2
2		SEQ1:DEL:ALL
2		SEQ1:ADD #h4012,2048,0,0
3		SEQ1:BEG RECEIVE,65536,1
4		$ apid = 2;
4		$ apnum = lp_create_ap (fd, apid);
5		$ chnum = lp_wait_for_chan (fd, apnum, &transmitter_addr, &max_blk_size);

6	apid = 1;
	receiver_addr.a_wwn.hi = 0x22222222;
	receiver_addr.a_wwn.lo = 0x00000002;
	receiver_addr.a_apid = 2;	

7	apnum = lp_create_ap(fd, apid);	
8	chnum = lp_setup_chan (fd, apnum, &receiver_addr, &max_blk_size);	
9		$ stat = lp_recv (fd, apnum, &buf, 2048, apnum, chnum, &numrec);
10	stat = lp_send (fd, &buf, 2048, apnum, chnum,&numsend);	
11	stat = lp_send (....	$ stat = lp_recv (...
12	stat = lp_close_chan (fd, apnum,chnum);	
13		$ stat = lp_close_chan (fd, apnum, chnum);
14	stat = lp_destroy_ap(fd, apnum);	$ stat = lp_destroy_ap(fd, apnum);
15		

Notes

1 The SCPI command `MMEM:SESS1:PROT` is sent to the HP E2749 to select the Systran Lightweight Protocol (LWSYSTRAN), and identify the APID of the receiver. Since the HP E2749A is the receiving application the `<receiver WWN>` parameter values don't need to be specified.

2 Load a sequence into the HP E2749A. For this example the sequence operation `#h4012` is used to receive bytes from Fibre Channel an put them into shared RAM. `2048` is the number of bytes to receive each time the sequence operation is executed.

3 Start receiving 65536 bytes of data using Sequence 1 and Session 1. This SCPI command results in internal firmware calls to the FXLP API functions described in the steps that follow.

4 The HP E2749A has been instructed to receive data using application id (APID) with a value of 2 (from the `MMEM:SESS:PROT` command parameters). For the FXLP protocol to operate properly, the data receiving application should be started before the transmitting application.

5 The HP E2749A waits for a transmitter (e.g. the computer) to establish a connection. The address of the transmitter that connected is returned in `transmitter_addr`.

6 This step sets variables that specify which receiving APID at which WWN we are to connect to. For this example, assume that the WWN of the receiving HP E2749A module is `2222222200000002`.

7 Set up an APID for the transmitter.

8 Connect to the receiving APID (2) at the specified WWN — in this example the WWN of the receiving HP E2749A is assumed to be `2222222200000002`.

9 The computer's call to `lp_setup_chan()` causes the HP E2749A's internal firmware call to `lp_wait_for_chan()` to unblock. At this time the HP E2749A knows that a transmitting node (the computer) has established a connection.

10 The HP E2749A calls `lp_recv()` multiple times to receive the 65536 bytes, as specified in the `SEQ:BEG` command, from the computer.

11 The computer calls `lp_send()` multiple times to transfer the 65536 bytes.

12 The computer calls `lp_close_chan()`. This causes the HP E2749A's call to `lp_recv()` to return with an indication that the channel has been closed by the transmitter.

13 The HP E2749A firmware calls `lp_close_chan()`.

14 The HP E2749A firmware and the computer call `lp_destroy_ap()`.

15 At this point 65536 bytes have been transferred from the computer to the HP E2749A.

Glossary

A16
16-bit address space. A16 has an upper limit of 65535.

A24
24-bit address space. A24 has an upper limit of 16777215.

A32
32-bit address space. A32 has an upper limit of 4294967295.

ADC
an Analog-to-Digital Converter module used as the input to a VXI system. Examples include the HP E1413B and HP E1431A.

address space
a range of addresses in memory. See also A16, A24, A32, and Shared RAM.

backplane
a set of lines that connects all the modules in a VXI system.

bit bucket
a place to put unwanted data.

D16
a single 16-bit transfer over the VXI system bus.

D32
a single 32-bit transfer over the VXI system bus.

embedded computer
a computer (functioning as controller) which is installed in the VXI mainframe. An example is the HP V743.

FC-PH
Fibre Channel Physical standard, which includes the lowest three levels, FC-0, FC-1, and FC-2.

FC-0
defines the physical characteristics of the media, including cables, connectors, drivers, transmitters, transmission rates, receivers, and optical and electrical parameters.

FC-1
defines the transmission protocol, including the 8B/10B encode/decode scheme, byte synchronization, and character-level error control.

FC-2
defines the signaling protocol, including the frame and byte structure, which is the data transport mechanism used by Fibre Channel.

FC-3
defines a set of services that are common across multiple ports of a node (still being formulated).

FC-4
defines the mapping of the upper level protocol interfaces to Fibre Channel constructs.

Fibre Channel
flexible, scaleable, high-speed data transfer interface that can operate over a variety of both copper wire and optical fiber. Networking and I/O protocols are mapped to Fibre Channel constructs, encapsulated, and transported within Fibre Channel frames.

frame
the smallest transfer unit for Fibre Channel.

FXLP
Systran Corporation's FibreXpress Lightweight protocol.

gigabit link module
a PC board and connector that provides the interface between the HP E2749 and the Fibre Channel. The connectors are either coaxial copper or optical.

GLM
see gigabit link module.

HP-IB
Hewlett-Packard Interface Bus.

implied mnemonic
keywords in a SCPI command which can be deleted without changing the effect of the command. Implied mnemonics are identified by brackets [ ] in SCPI syntax diagrams.

LBUS
see Local Bus.

Local Bus
a high-speed port that Hewlett-Packard had defined as a standard byte-wide ECL protocol which can transfer measurement data from left to right at up to 2.62 Msamples per second on the VXI backplane.

logical address
the VXI address of a module.

memory space
see address space.

monitoring
a method of transferring data which allows the host computer to access part of the data during transfer operations. This is done by transferring part of the data to host memory at the same time as to the HP E2749 Session.

MXI
an interface to extend the VXI bus to the memory space of a host computer.

primary address
one of 3 parts of HP E2749 address in a SCPI environment. The primary address, typically 09, indicates which HP-IB port in the system controller is used to communicate with the Slot 0 Control Module, for example the HP E1406A/B.

protocol
A data transmission convention, including timing, control, formatting, and data representation.

SCPI
Standard Commands for Programmable Instruments, a standard instrument command language.

secondary address
one of 3 parts of HP E2749 address in a SCPI environment. The secondary address indicates the device-specific address. In this case, the VXI logical address.

select code
one of 3 parts of HP E2749 address in a SCPI environment. The select code specifies the interface. Seven (7) is a typical number for the HP-IB interface.

sequence
specifies the order of operations for transmitting or receiving data.

shared memory
see Shared RAM.

Shared RAM
memory space that is available to be shared with other devices, as a way of passing data. Shared RAM has an upper limit of 262143. (RAM = Random Access Memory).

SRQ
Service Request.

system bus
a way of referring to the VXI bus not including the Local Bus.

TTLTRG
eight lines on the VXI backplane which are available to provide synchronization between devices. The HP E2749 uses the TTLTRG lines for simple communication with other devices.

VXI
VME Extensions for Instrumentation, a standard specification for instrument systems.

VXI*plug&play*
a set of standards which provides VXI users with a level of standardization across different vendors beyond what the VXI standard specifications spell out.

INDEX

Q

R

S

T

U

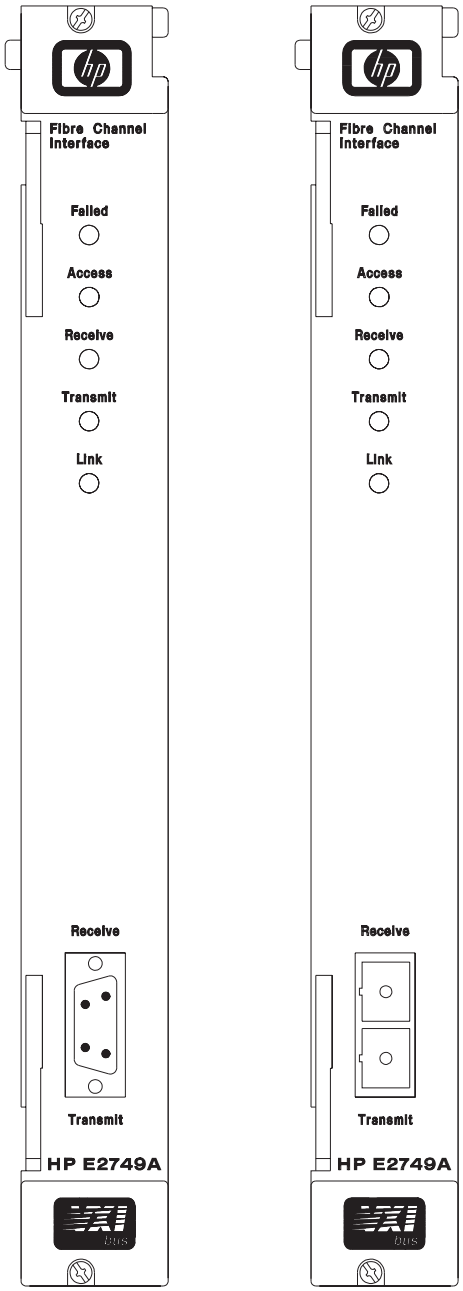
V

W

HP E2749A

Technical Specifications

HP E2749A
Fibre Channel
Interface Module



The HP E2749A Fibre Channel Interface Module provides high speed data transfer to and from VXI mainframes. Transfer data at rates up to 41 Mbytes/sec depending on the system configuration and your application. Compare this to typical VXIbus and MXIbus transfer rates of 2 to 4 Mbytes/sec.

The HP E2749A is compatible with Systran Corporation's FibreXpress™ Lightweight Protocol (FXLP).

Specifications describe warranted performance. **Benchmarks** provide useful information indicative of performance of product in typical operational scenarios. The actual performance in a given application will depend on the actual hardware and software used.

Specifications

General Specifications

Emissions, Rad & Cond	CISPR 11, Group 1, Class A
Immunity standards	
IEC 1000-2-3 ESD	8 kVAD, 4 kVCD
IEC 1000-2-3 Radiated immunity	3 V/m
IEC 1000-2-4 Fast transients	I/O 500 V
Safety	Option 001 includes a class 1 laser device per IEL 825-1. Device is eye safe. Complies with 21 Code of Federal Regulations (CFR, USA), chapter 1, subpart J.
GLM interface	The GLM used in the HP E2749 meets the industry standard defined within the ANSI X3.230-1994 and X3T11/Project 1119D/Rev 8.3 standards, and is FCSI-301-Rev 1.0 compatible.
VXI bus standards	VXI (Rev. 1.4); Message-based servant; A16/A24/A32, D08/D16/D32 Master; A16/A24, D08/D16 Slave Interrupter/handler

VXI power requirement	**dc**	**dynamic current**
+5 V	3.7A	0.60 A
— 5.2 V	0.6 A	0.01 A
— 2 V	0.14 A	0.07 A
+12 V	0.00 A†	0.00 A
— 12 V	0.00 A	0.00 A
+24 V	0.00 A	0.00 A
— 24 V	0.00 A	0.00 A
+5 V Standby	0.00 A	0.00 A

† The HP E2749A will consume up to 60 mA (20 mA typ) of +12 while programming ROMS.

VXI cooling requirement (10°C rise)	1.75 liter/second, 0.1 mm H_2O
Weight	
Net	1.1 kg (2.5 lbs)
Shipping	4.3 kg (9.4 lbs)
Dimensions	Single slot, C-size VXI module
Maximum cable length	
Optical (850 nm multimode laser, 50µm core diameter)	500 m
Coaxial copper (150 Ω STP connector)	30 m

Environmental

Operating restrictions	
Ambient temperature	0° to 60°C
Humidity, non-condensing	< 80% RH at 40°C
Maximum altitude	2300 meters (7,500 feet)
Storage and transport restrictions	
Ambient temperature	– 40° to 75°C
Humidity, non-condensing	5% RH, 95% RH at 65°C
Maximum altitude	4600 meters (15,000 feet)

Performance Benchmarks

Maximum data transmit rate	41 Mbytes/sec
Maximum data receive rate	***TBD*** - Gary
need some examples???	

HEWLETT® PACKARD

For more information on Hewlett-Packard Test & Measurement products, applications or services please call your local Hewlett-Packard sales offices. A current listing is available via Web through Access HP at http://www.hp.com. If you do not have access to the internet please contact one of the HP centers listed below and they will direct you to your nearest HP representative.

United States:

Hewlett-Packard Company
Test and Measurement Call Center
P.O. Box 4026
Englewood, CO 80155-4026

Canada:

Hewlett-Packard Canada Ltd.
5150 Spectrum Way
Mississauga, Ontario
L4W 5G1
(905) 206 4725

Europe:

Hewlett-Packard
European Marketing Centre
P.O. Box 999
1180 AZ Amstelveen
The Netherlands
(31-20) 547-9900

Japan:

Hewlett-Packard Japan Ltd.
Measurement Assistance Center
9-1, Takakura-Cho, Hachioji-Shi,
Tokyo 192, Japan
Tel: (81-426) 56-7832
Fax: (81-426) 56-7840

Latin America:

Hewlett-Packard
Latin American Region Headquarters
5200 Blue Lagoon Drive
9th Floor
Miami, Florida 33126
U.S.A.
(305) 267 4245/4220

Australia/New Zealand:

Hewlett-Packard Australia Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
Australia
1 800 629 485

Asia Pacific:

Hewlett-Packard Asia Pacific Ltd.
17-21/F Shell Tower, Times Square,
1 Matheson Street, Causeway Bay,
Hong Kong
Tel: (852) 2599-7777
Fax: (852) 2506-9285

Printed in USA 6/97
????-????E

Declaration of Conformity

According to ISO/IEC Guide 22 and EN 45014

Manufacturer's name:	Hewlett-Packard Company
Manufacturer's address:	Lake Stevens Instrument Division 8600 Soper Hill Road Everett, Washington 98205-1298

declares, that the product

Product Name:	Fibre Channel Interface Module
Model Number:	HP E2749A

conforms to the following specifications:

Safety:	IEC 1010-1:1990 + A1/EN61010:1993
EMC:	CISPR 11: 1990/EN55011 (1991), Group1, Class A IEC 801-2: 1991/EN50082-1 (1992): 4 kV CD, 8 kV AD IEC 801-3: 1984/EN50082-1 (1992): 3 V/m (1) IEC 801-4: 1988/EN50082-1 (1992): 1 kV

Supplementary Information:

The product herewith complies with the requirements of the Low Voltage Directive 73/23/EEC and the EMC Directive 89/336/EEC.

(1) In a 3 V/m field, some degradation of product performance occurs.

Everett, Washington - Sept. 19, 1996

Cathy Thran, Quality Manager

Need Assistance?

If you need assistance, contact your nearest Hewlett-Packard Service Office listed in the HP Catalog, or contact your nearest regional office listed at the back of this book. If you are contacting Hewlett-Packard about a problem with your HP E2749 Fibre Channel Interface Module, please provide the following information:

- ❑ Model number: HP E2749A
- ❑ Software version:
- ❑ Serial number:
- ❑ Options:
- ❑ Date the problem was first encountered:
- ❑ Circumstances in which the problem was encountered:
- ❑ Can you reproduce the problem?
- ❑ What effect does this problem have on you?

About this edition

June 1997: First Edition. The diagnostic programs were designed for firmware version A.00.00.