Contents

Notes

Certification

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

Warranty

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 Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other HP products.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

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

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

Limitation Of Warranty

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

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

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

Exclusive Remedies

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

Notice

The information contained in this document is subject to change without notice. HEWLETT-PACKARD (HP) MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HP shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. HP assumes no responsibility for the use or reliability of its software on equipment that is not furnished by HP.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19 (c) (1,2).

HP E1463A 32-Channel, 5 Amp, Form C Switch Module User's Manual
Edition 3

Documentation History

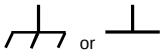
All Editions and Updates of this manual and their creation date are listed below. The first Edition of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct or add additional information to the current Edition of the manual. Whenever a new Edition is created, it will contain all of the Update information for the previous Edition. Each new Edition or Update also includes a revised copy of this documentation history page.

Edition 1 . . . . . July 1991
Edition 2 . . . . . January 1994
Edition 3 . . . . . January 1996

Safety Symbols

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

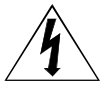
Indicates the field wiring terminal that must be connected to earth ground before operating the equipment—protects against electrical shock in case of fault.



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

Alternating current (AC).

Direct current (DC).



Indicates hazardous voltages.

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

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

WARNINGS

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

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

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

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

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

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

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

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

Declaration of Conformity

according to ISO/IEC Guide 22 and EN 45014

Manufacturer's Name:	Hewlett-Packard Company Loveland Manufacturing Center
Manufacturer's Address:	815 14th Street S.W. Loveland, Colorado 80537

declares, that the product:

Product Name:	32-Channel Form C Switch
Model Number:	E1463A
Product Options:	All

conforms to the following Product Specifications:

Safety:	IEC 1010-1 (1990) Incl. Amend 1 (1992)/EN61010-1 (1993) CSA C22.2 #1010.1 (1992) UL 3111
EMC:	CISPR 11:1990/EN55011 (1991): Group1 Class A IEC 801-2:1991/EN50082-1 (1992): 4kVCD, 8kVAD IEC 801-3:1984/EN50082-1 (1992): 3 V/m IEC 801-4:1988/EN50082-1 (1992): 1kV Power Line

Supplementary Information: The product herewith complies with the requirements of the Low Voltage Directive 73/23/EEC and the EMC Directive 89/336/EEC (inclusive 93/68/EEC) and carries the "CE" mark accordingly.

Tested in a typical configuration in an HP C-Size VXI mainframe.

August 20, 1995 **Jim White, QA Manager**

European contact: Your local Hewlett-Packard Sales and Service Office or Hewlett-Packard GmbH, Department HQ-TRE, Herrenberger Straße 130, D-71034 Böblingen, Germany (FAX +49-7031-14-3143).

Notes

Please fold and tape for mailing

Reader Comment Sheet

HP E1463A 32-Channel, 5 Amp, Form C Switch User's Manual
Edition 3

You can help us improve our manuals by sharing your comments and suggestions. **In appreciation of your time, we will enter you in a quarterly drawing for a Hewlett-Packard Palmtop Personal Computer** (U.S. government employees cannot participate in the drawing).

Your Name	City, State/Province
Company Name	Country
Job Title	Zip/Postal Code
Address	Telephone Number with Area Code

Please list the system controller, operating system, programming language, and plug-in modules you are using.

fold here

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 37 LOVELAND, CO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD COMPANY
Measurement Systems Division
Learning Products Department
P.O. Box 301
Loveland, CO 80539-9984

fold here

cut along this line

Please pencil-in one circle for each statement below:

	Disagree ←				→ Agree
• The documentation is well organized.	O	O	O	O	O
• Instructions are easy to understand.	O	O	O	O	O
• The documentation is clearly written.	O	O	O	O	O
• Examples are clear and useful.	O	O	O	O	O
• Illustrations are clear and helpful.	O	O	O	O	O
• The documentation meets my overall expectations.	O	O	O	O	O

Please write any comments or suggestions below--be specific.

Chapter 1
Getting Started with the HP E1463A Form C Switch

Using This Chapter

This chapter includes a Form C switch description, addressing guidelines, configuration information, and an example program to check initial operation. Chapter contents are as follows:

Instrument Definition

HP plug-in modules installed in an HP mainframe or used with an HP command module are treated as independent instruments each having a unique secondary HP-IB address. Each instrument is also assigned a dedicated error queue, input and output buffers, status registers and, if applicable, dedicated mainframe/command module memory space for readings or data. An instrument may be composed of a single plug-in module (such as a counter) or multiple plug-in modules (for a switchbox or scanning multimeter instrument).

Form C Switch Description

The HP E1463A 32-Channel, 5 Amp, Form C Switch Module is a C-Size VXIbus and VMEbus register-based product which can be used for switching, scanning, and control. The switch can operate in a C-Size VXIbus or VMEbus mainframe. The switch has 32 channels of Form C relays. Each channel includes a relay with common (C), normally open (NO), and normally closed (NC) contacts.

For the Form C switch, switching consists of opening or closing a channel relay to provide alternate connections to user devices. Scanning consists of closing a set of relays, one at a time.

Basic Operation

As shown in Figure 1-1, the Form C switch module consists of 32 channels (channels 00 through 31). Each channel uses a nonlatching relay. Varistors (MOVs) can be added for relay protection, and resistors or fuses can be added for circuit protection. See "Adding Relay and Circuit Protection" on page 25 for more information. External pull-up resistors can also be added for digital output applications. See pages 14 and 37 for additional information about these applications.

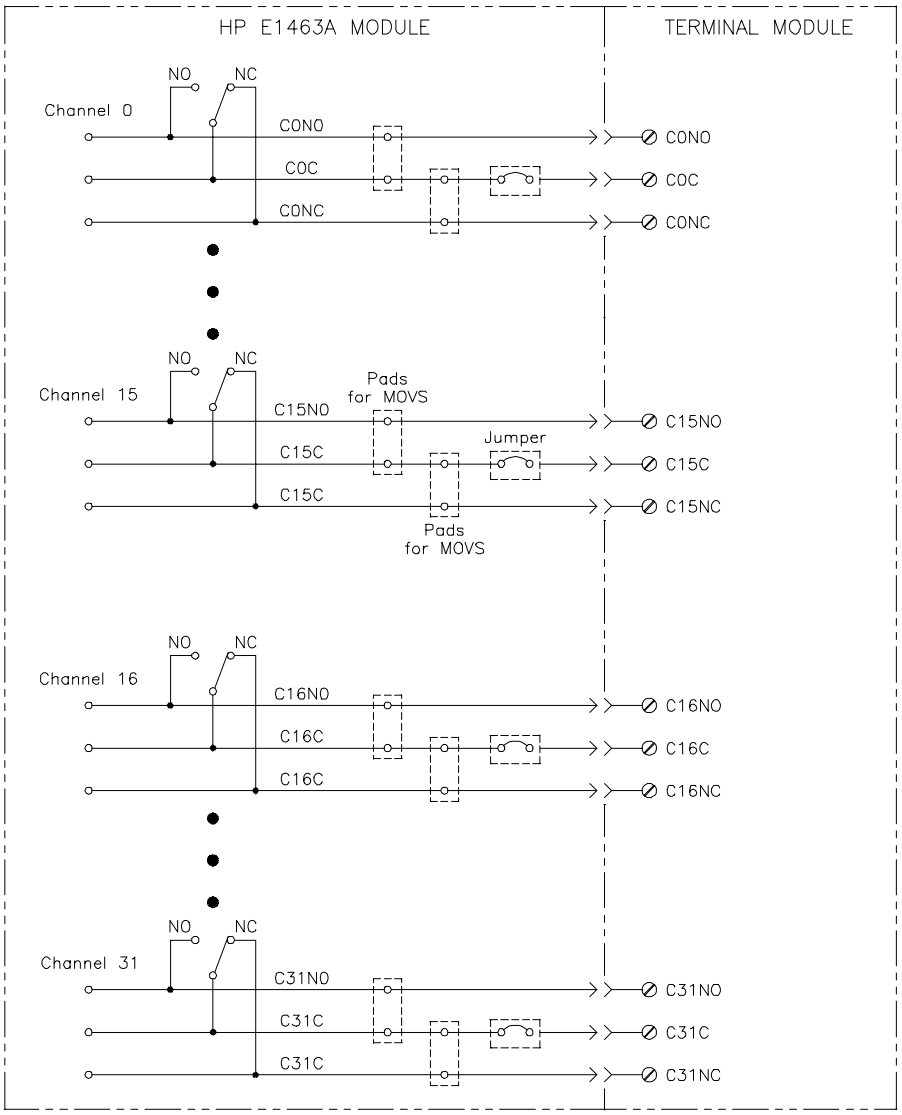


Figure 1-1. Form C Simplified Schematic

Each channel is switched by opening or closing the appropriate channel relay. Since the relays are nonlatching, the relays are all open during power-up or power-down. When a reset occurs, all channel commons (C) are connected to the corresponding normally closed (NC) contacts. When a channel is closed, the common contact (C) is connected to the normally open contact (NO). User inputs and outputs to each channel are via the NO, NC, and C terminal connectors on the terminal module.

Typical Configuration

The Form C switch accepts user inputs up to 125 Vdc or 250 Vac_{rms}. Maximum rated power capacity (external load) is 150 Wdc or 1250 VA per channel. Per module, you can switch 1500 Wdc or 12500 VA.

As noted, the switch may be configured for general purpose switching/ scanning or digital output applications. For general purpose switching or scanning, no additional configuration is required. To configure the switch for digital output applications, install external pull-up resistors as required.

Multiple Form C switch modules can be configured as a switchbox instrument. When using a switchbox instrument, multiple Form C switch modules within the switchbox instrument can be addressed using a single interface address. This configuration, however, requires the use of SCPI (Standard Commands for Programmable Instruments) commands. SCPI commands are discussed throughout this manual and information on the switchbox instrument configuration can be found in the *C-Size VXIbus System Installation and Getting Started Guide*.

Warnings and Cautions

WARNING **SHOCK HAZARD. Only qualified, service-trained personnel who are aware of the hazards involved should install, configure, or remove the Form C switch module. Use only wire rated for the highest input voltage and remove all power sources from the mainframe and installed modules before installing or removing a module.**

CAUTION **MAXIMUM VOLTAGE/CURRENT.** Maximum allowable voltage per channel for the Form C switch is 125 Vdc or 250 Vac_{rms}. Maximum current per channel is 5 Adc or ac_{rms} (non-inductive). Maximum power of an external load is 150 Wdc or 1250 VA per channel or 1500 Wdc or 12500 VA per module. Exceeding any limit may damage the Form C switch.

STATIC ELECTRICITY. Static electricity is a major cause of component failure. To prevent damage to the electrical components in the Form C switch, observe anti-static techniques whenever removing a module from the mainframe or whenever working on a module. The Form C switch is susceptible to static discharges. Do not install the Form C switch module without its metal shield attached.

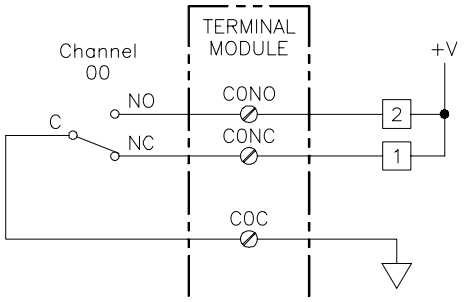
Configuring the Form C Switch

Typical Form C switch configurations are as a general purpose relay or digital output.

General Purpose Relay Configuration

As factory-configured, the Form C switch module is set for general purpose relay configuration. For this configuration, you can switch channels by opening or closing channel relays or you can scan a set of channels.

Figure 1-2 shows a typical general purpose relay configuration for channel 00. When the relay is open (NC terminal is connected to the C terminal), load 1 is connected. When the relay is closed (NO terminal is connected to the C terminal), load 2 is connected.



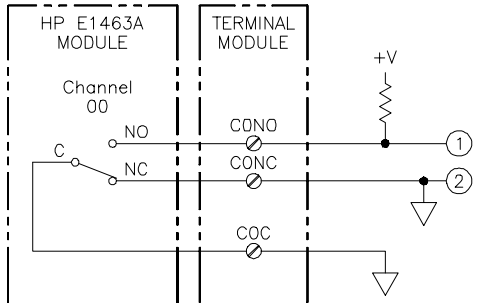
Relay	Load
Open	1
Closed	2

Figure 1-2. General Purpose Relay Configuration

Digital Output Configuration

By installing external pull-up resistors, the Form C switch can be configured as a digital output device.

Figure 1-3 shows channel 00 configured for digital output operation. When the channel 00 relay is open (NC connected to C), point 1 is at + V. When the channel 00 relay is closed (NO connected to C), point 1 is at 0 V.



Relay	①	②
Open	+V	0V
Closed	0V	0V

Figure 1-3. Digital Output Configuration

Setting the Logical Address Switch

The logical address switch (LADDR) factory setting is 120. Valid addresses are from 1 to 255. The Form C switch module can be configured as a single instrument or as a switchbox. Refer to the *C-Size VXIbus System Installation and Getting Started Guide* for addressing information. Refer to Figure 1-4 for switch position information.

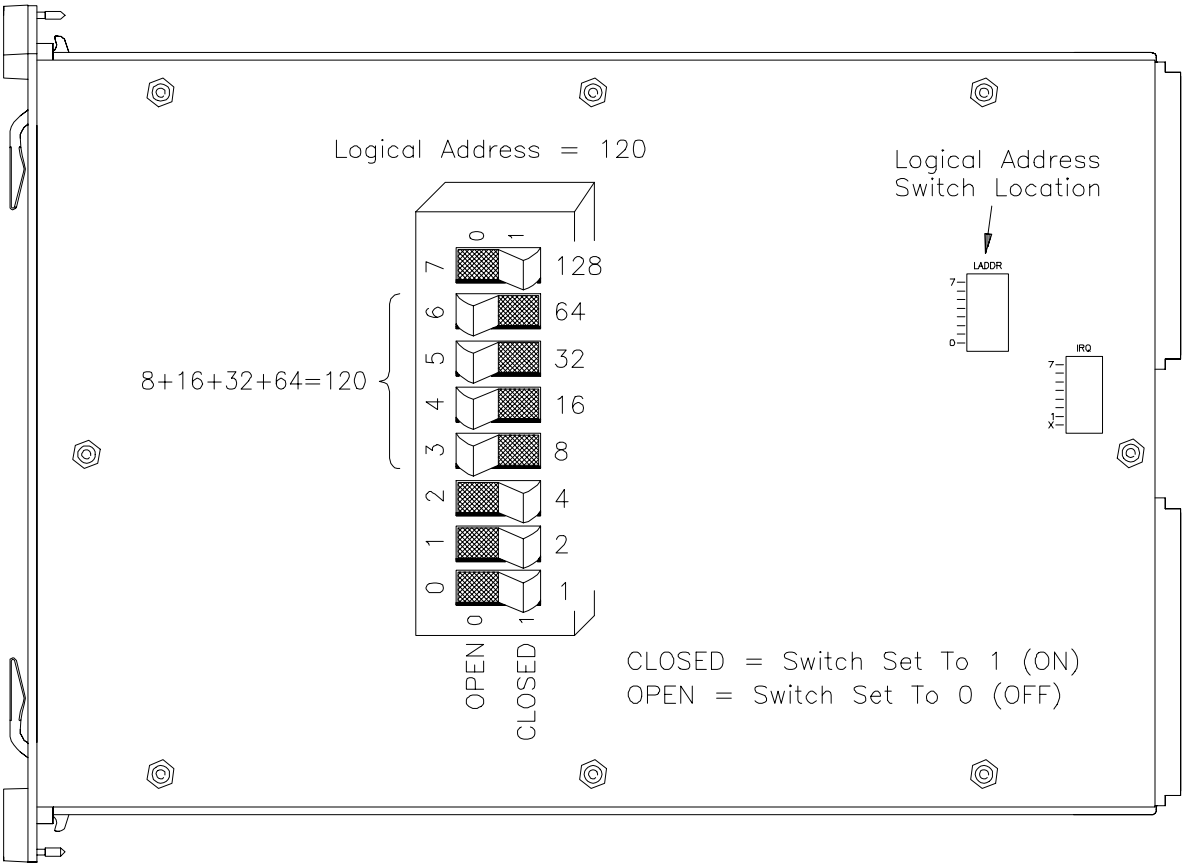


Figure 1-4. Setting the Logical Address Switch

Note

The address switch selected value must be a multiple of 8 if the module is the first module in a switchbox used with a VXIbus command module, and being instructed by SCPI commands.

Setting the Interrupt Priority

The Form C switch module generates an interrupt after a channel has been closed. These interrupts are sent to, and acknowledgments are received from, the command module (HP E1406A, for example) via the VXIbus backplane interrupt lines.

For most applications where the Form C switch module is installed in an HP 75000 Series C mainframe, the interrupt priority jumper does not have to be moved. This is because the VXIbus interrupt lines have the same priority, and interrupt priority is established by installing modules in slots numerically closest to the command module. Thus, slot 1 has a higher priority than slot 2, slot 2 has a higher priority than slot 3, and so on.

Refer to Figure 1-5 to change the interrupt priority. You can select eight different interrupt priority levels. Level 1 is the lowest priority and Level 7 is the highest priority. Level X disables the interrupt. The module's factory setting is Level 1. To change, remove the 4-pin jumper (HP P/N 1258-0247) from the old priority location and reinstall in the new priority location. If the 4-pin jumper is not used, the two jumper locations must have the same interrupt priority level selected.

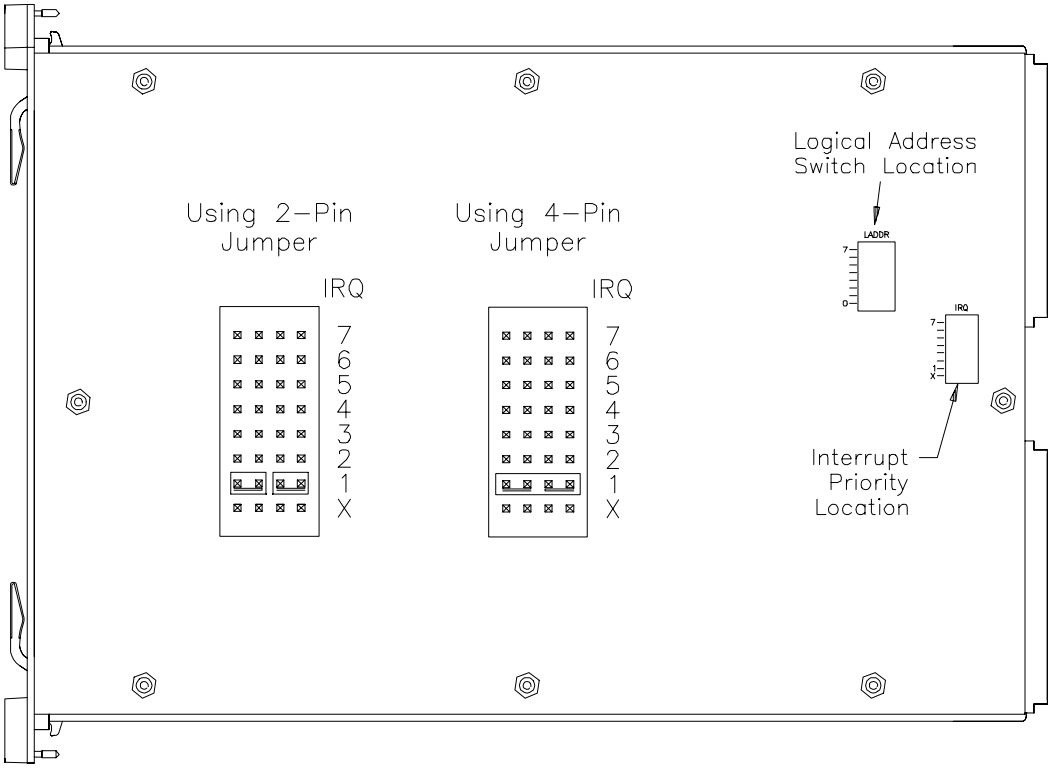


Figure 1-5. Setting the Interrupt Line

Note

The interrupt priority jumper **MUST** be installed in position 1 when using the HP E1405/06A command module. Level X interrupt priority should not be used under normal operating conditions. Changing the priority level jumper is not recommended. Do not change unless specifically instructed to do so.

Installing the Form C Switch in a Mainframe

The HP E1463A may be installed in any slot (except slot 0) in a C-size VXIbus mainframe. Refer to Figure 1-6 to install the Form C switch in a mainframe.

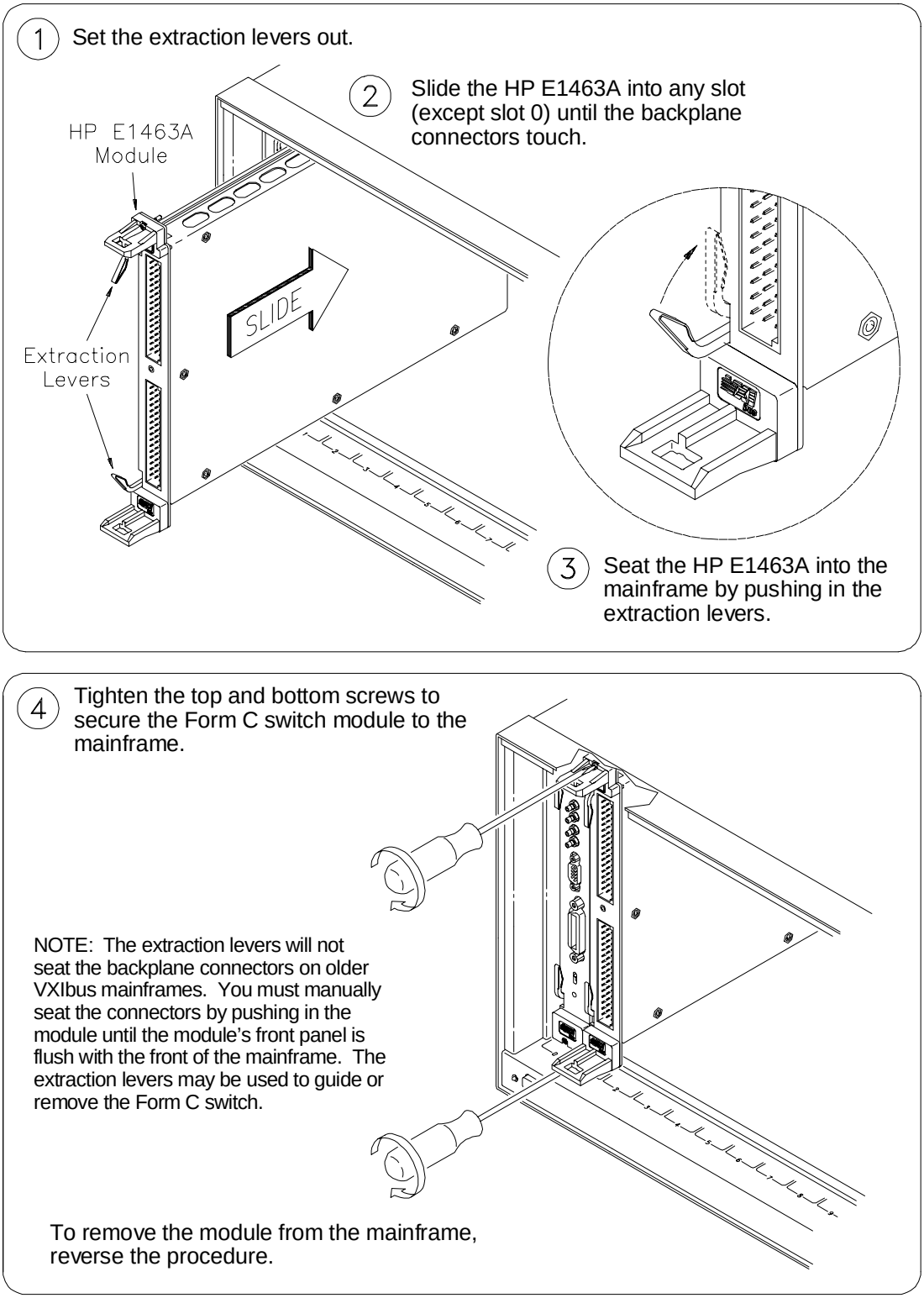


Figure 1-6. Installing the Form C Switch in a VXIbus Mainframe

Terminal Modules

The HP E1463A 32-Channel, 5 Amp, Form C Switch Module is comprised of a relay switch card and a screw type standard terminal module. User inputs to the Form C switch are to the normally open (NO), normally closed (NC), and common (C) terminal connectors on the screw type terminal module. If the screw type terminal module is not desired, a solder eye terminal module (Option A3G) is available. If the solder eye terminal module without the HP E1463A Form C switch card is desired, order HP part number E1463-80012. See Figure 1-9 on page 20 for the Form C switch connector pin-out which mates to the terminal module.

Screw Type Terminal Module

Figure 1-7 shows the HP E1463A's standard screw type terminal module connectors and associated channel numbers. Use the guidelines below for wire connections.

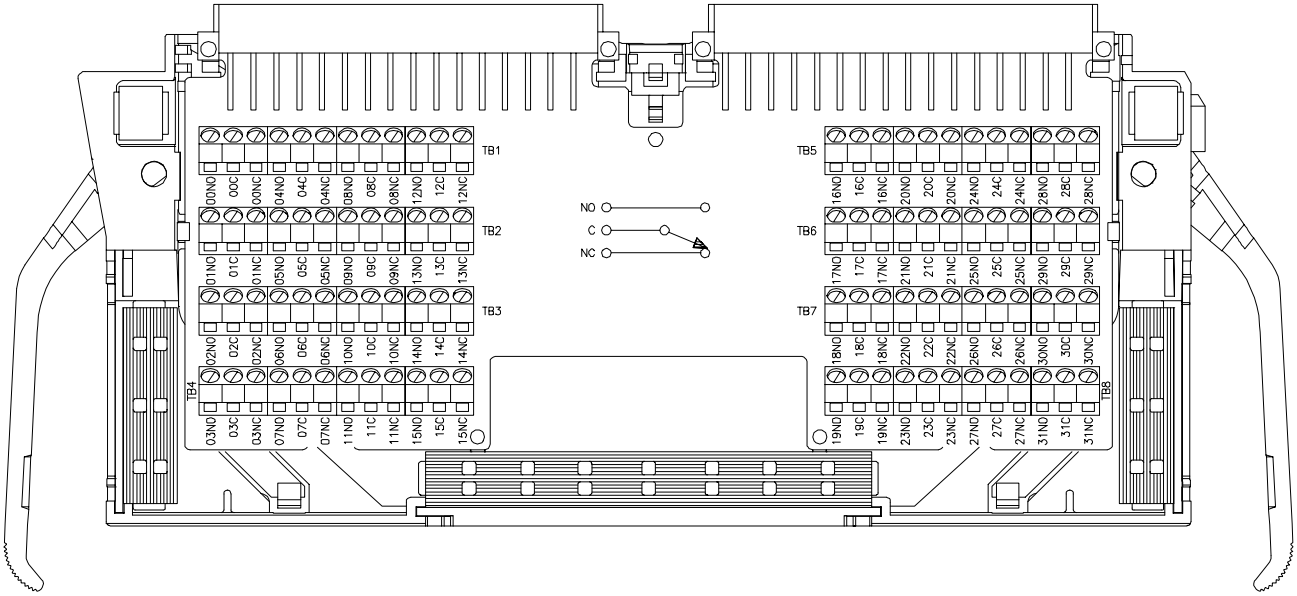


Figure 1-7. HP E1463A Standard Screw Type Terminal Module

Wiring Guidelines

- Be sure the wires make good connections on screw terminals.
- Maximum terminal wire size is No. 16 AWG. When wiring all channels, a smaller gauge wire (No. 20 - 22 AWG) is recommended. Wire ends should be stripped 6 mm (≈ 0.25 inch) and tinned to prevent single strands from shorting to adjacent terminals.

Note Refer to pages 22 and 23 before attempting to wire the terminal module.

Terminal Module Option A3G

A terminal module with screw type terminals is provided standard with the HP E1463A. Option A3G can be ordered if a solder eye terminal module is desired. Option A3G provides a plastic terminal module housing with solder eye connectors (see Figure 1-8). This allows you to solder wires onto connectors which are then inserted directly into the mating connector of the Form C switch. Use the pin-out diagram on page 20 to make the connections.

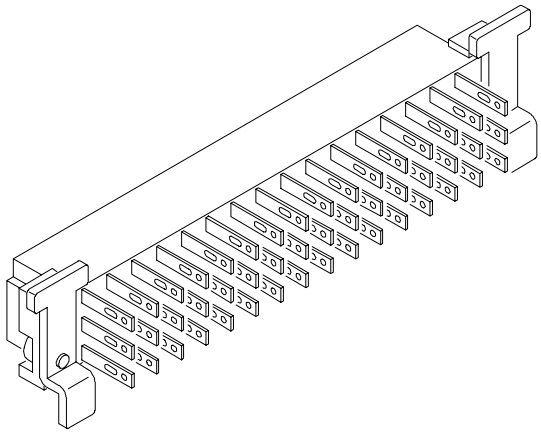


Figure 1-8. Solder Eye Connector

Connecting User Inputs

Figure 1-9 shows the front panel of the HP E1463A and the Form C switch's connector pin-out which mates to the terminal module.

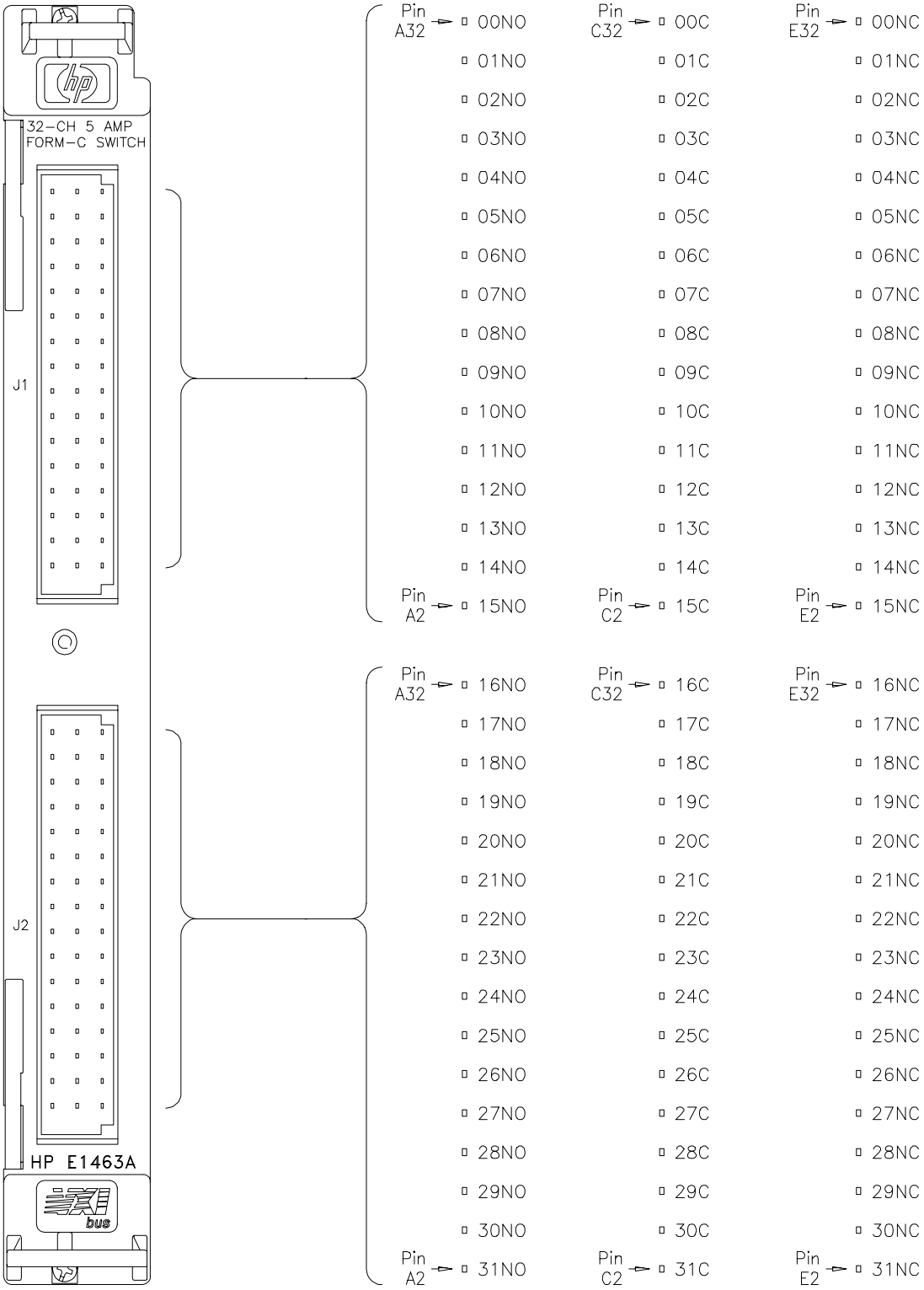
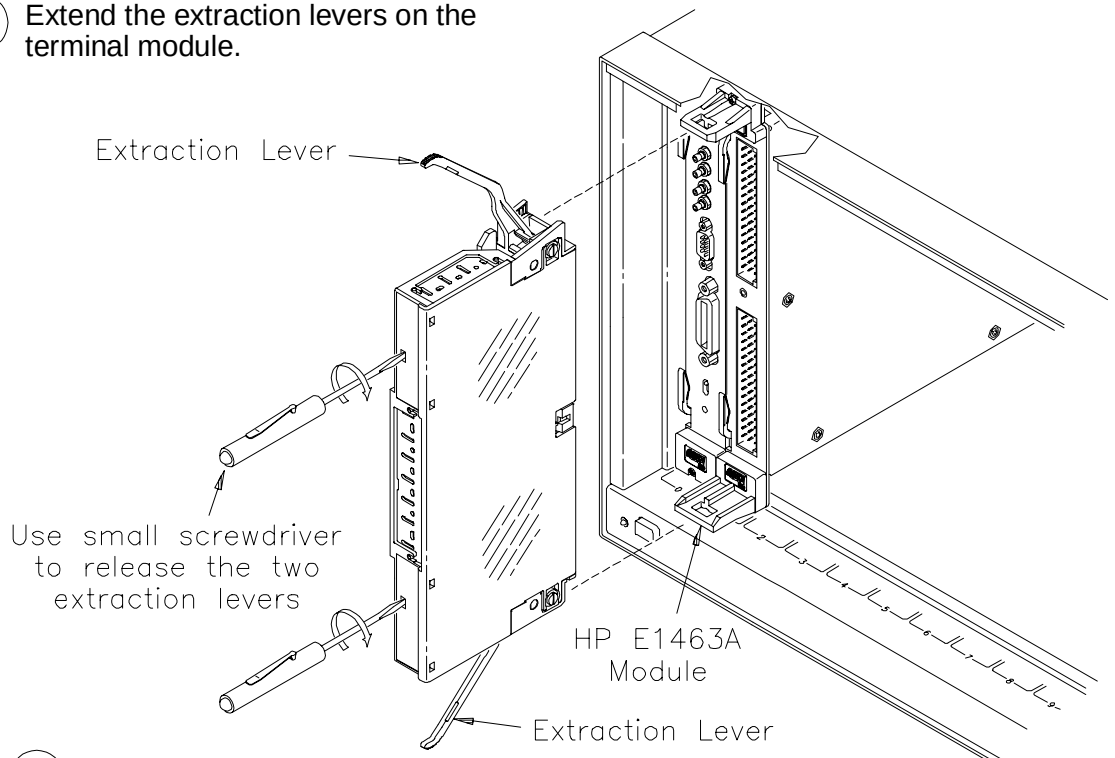


Figure 1-9. HP E1463A Form C Switch Pin-out

Attaching a Terminal Module to the Form C Switch

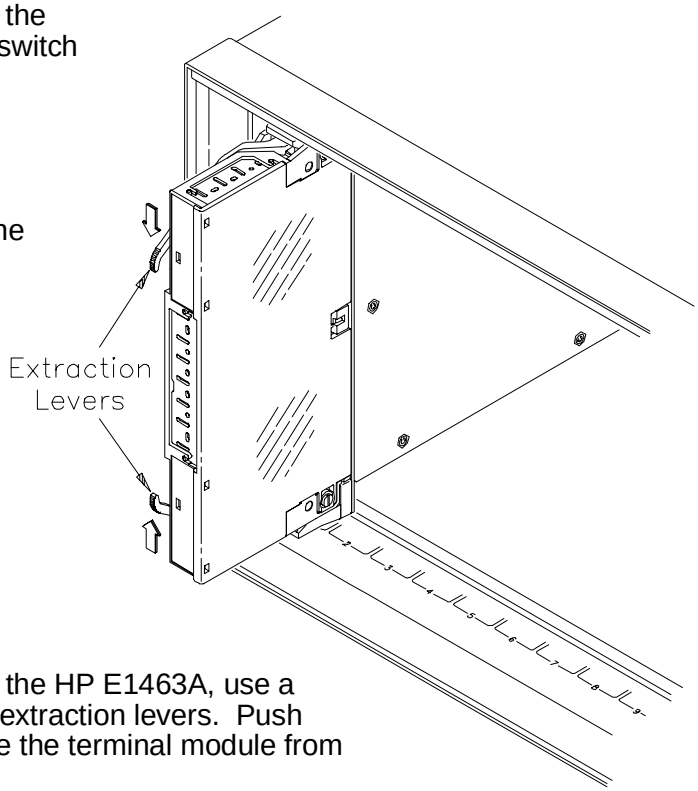
1. Extend the extraction levers on the terminal module.



2. Align the terminal module connectors to the Form C switch module connectors.

3. Apply gentle pressure to attach the terminal module to the Form C switch module.

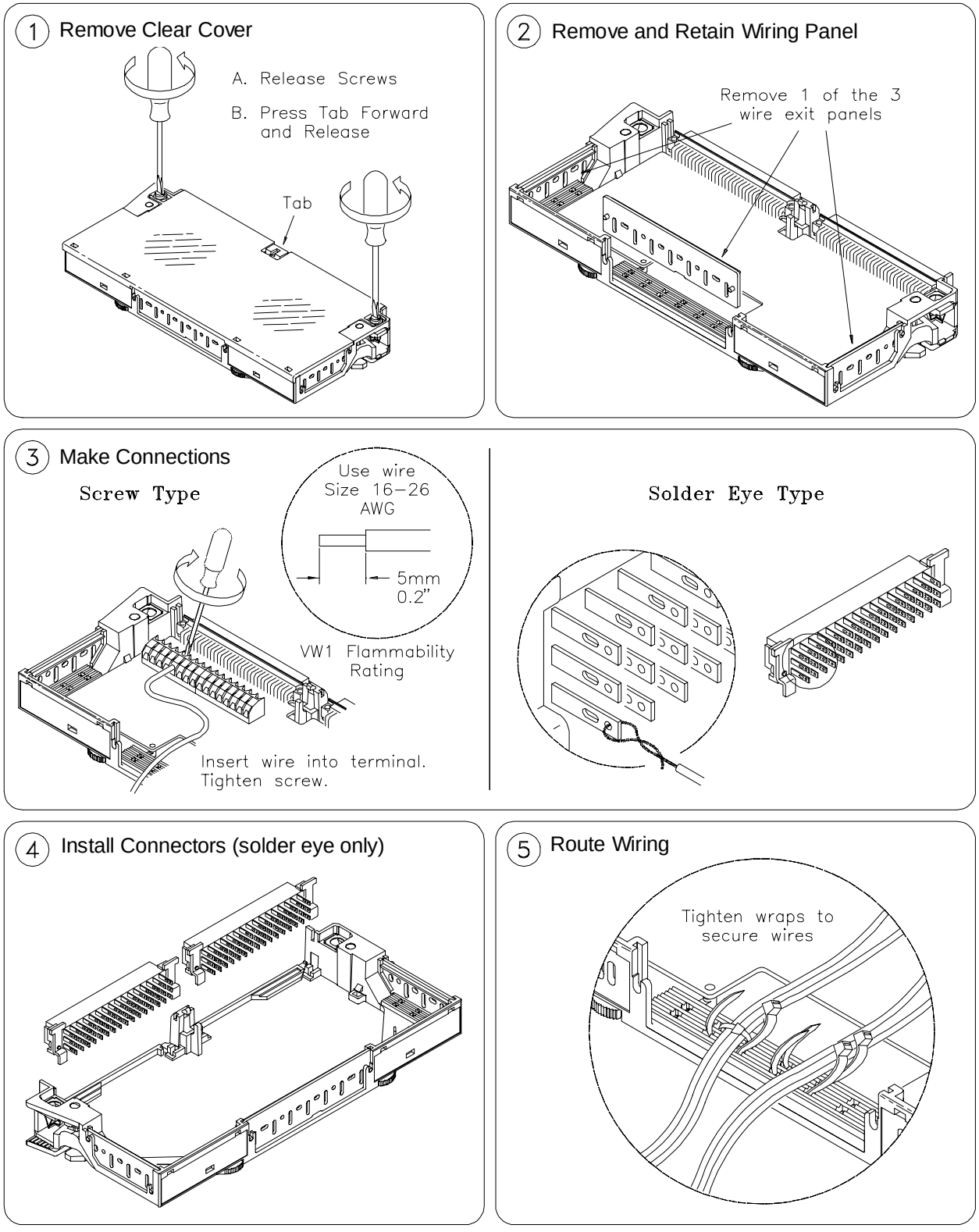
4. Push in the extraction levers to lock the terminal module onto the Form C switch module.



To remove the terminal module from the HP E1463A, use a small screwdriver to release the two extraction levers. Push both levers out simultaneously to free the terminal module from the Form C switch module.

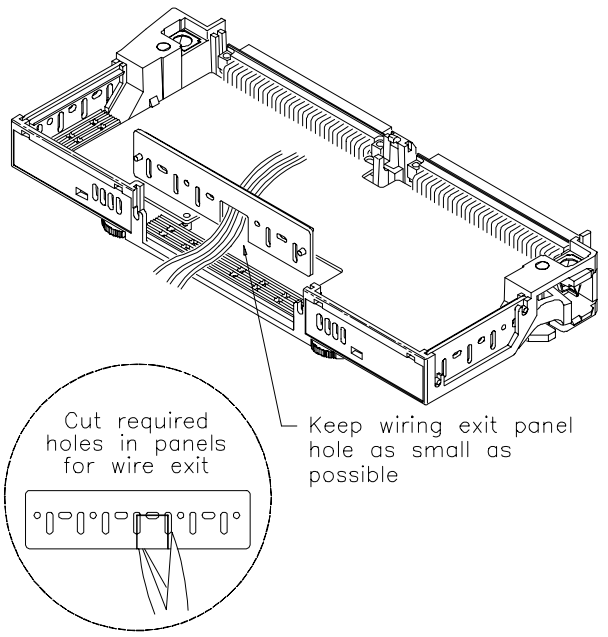
Wiring a Terminal Module

The following illustrations show how to connect field wiring to the terminal module.

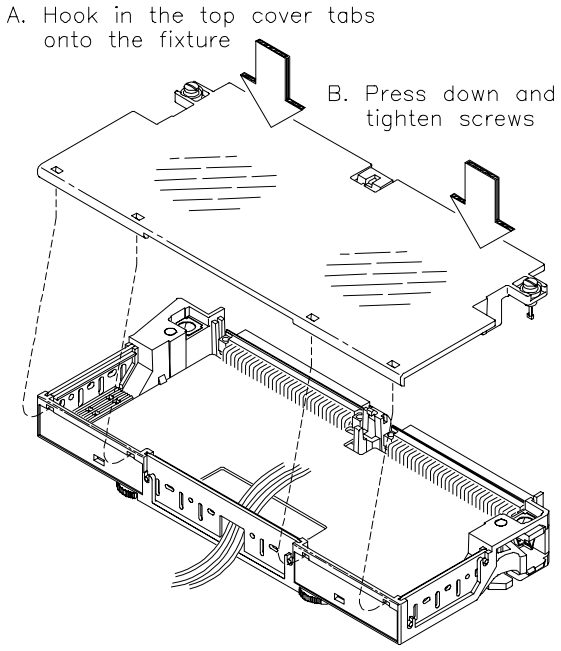


Continued on Next Page

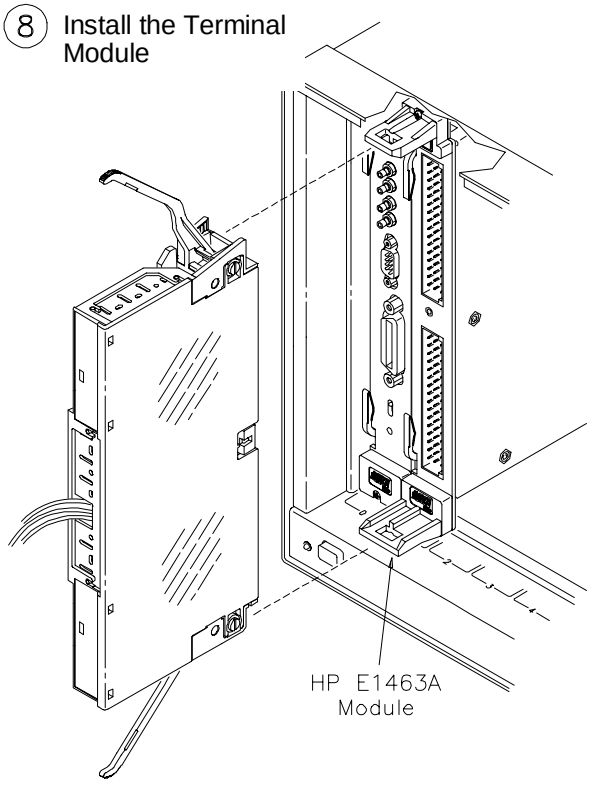
6 Replace Wiring Panel



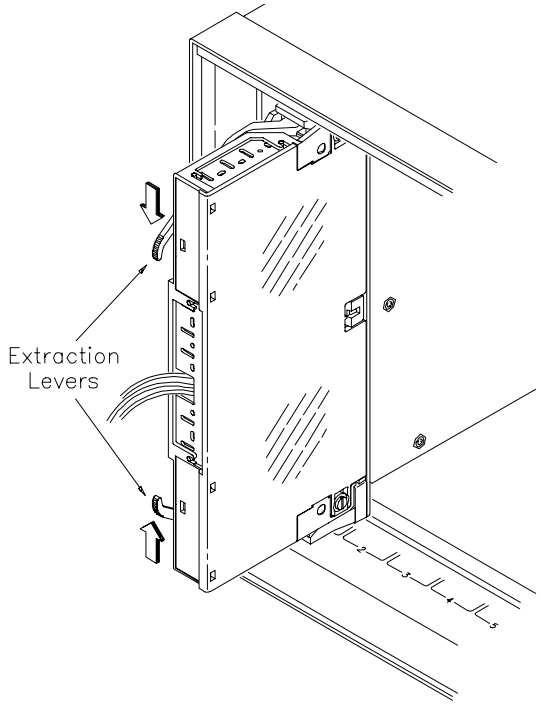
7 Replace Clear Cover



8 Install the Terminal Module



9 Push in the Extraction Levers to Lock the Terminal Module onto the HP E1463A



Protecting Relays and Circuits

Relays have a shorter life span than other electronic parts such as ICs. Because of their mechanical nature, relays usually have about 10 million operations (at 30 operations per second) which is not quite 100 hours. Therefore, to get the full life out of a relay in a switching module, you must protect the relay. The following sections provide additional information about protecting your relays and circuits.

Relay Reliability

To make sure you get the full life of your relays, keep the following in mind:

- **Be aware of non-resistive loads.** When switching inductive loads, high voltages (thousands of volts) are produced across the relay contacts. This causes arcing and transfer of material between contacts. Oxides and carbides from components of the atmosphere coat the contacts and cause high contact resistance. The transfer of material creates hills and valleys which lock together to "weld" contacts. Motor loads, for example, produce large inrush currents that can be 5 to 10 times greater than the steady state current. Table 1-1 summarizes how many times greater the inrush current can be for different types of loads.

Table 1-1. Inrush Currents

Type of Load	Inrush Current Times Steady State
Resistive	1
Solenoid	10 - 20
Motor	5 - 10
Incandescent Lamp	10 - 15
Mercury Lamp	3
Sodium Vapor Lamp	1 - 3
Capacitive	20 - 40
Transformer	5 - 15

- **Be aware of heavy current applications.** When a relay is used in heavy current applications, the thin layer of gold plating on the contact may be destroyed. This will not affect the heavy current application. If you go back to a low current application, however, you may experience a high contact resistance and be unable to use the relay for low current applications.

- **Use protective circuits with your relay connections.** The relay manufacturer recommends some protective circuits that can be used with your relay connections. Refer to the *Aromat Technical Data Book* (AGC-C0064-A-1) for additional information:

  Aromat
  401 River Oaks Parkway
  San Jose, CA 95134
  (408) 433-0466

Capacitors are not to be placed across the load or relay contacts. Capacitors may suppress arcs, but the energy stored in the capacitors will flow through the relay contacts, welding them.

You can also add circuit protection within the HP E1463A component module. See the sections following.

Adding Relay and Circuit Protection

Space has been made available on the HP E1463A Form C module for adding relay and circuit protection. Relay protection can be added by placing a protective device across the specified pads. This is done by adding metal oxide varistors (MOVs) between the common (C) and normally open (NO) or normally closed (NC) terminals. Now as the voltage goes up, the varistor draws current to protect the relay. Figure 1-10 shows the locations where these protective devices can be added.

Circuit protection can be added by placing a protective device in series with the common lead. This is done by adding a resistor or fuse between the common (C) terminal and your circuit. When installing circuit protection a jumper must be removed first.

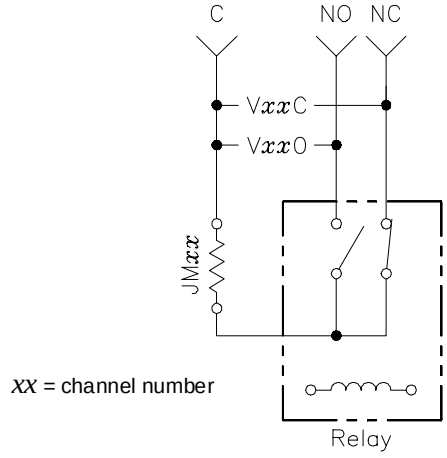


Figure 1-10. Adding Relay and Circuit Protection

To install these protective devices it is necessary to remove the sheet metal covers from the module. The locations for installing the devices are labeled as follows:

Table 1-2. Protective Devices Board Location

Relay Protection	Circuit Protection
V*xx*O	The varistor location across common (C) and normally open (NO).
V*xx*C	The varistor location across common (C) and normally closed (NC).
Circuit Protection	
JM*xx*	The resistor or fuse location in series with common (C).

Where *xx* is the channel number.

Again, do not install a capacitor in any of these locations.

Maximum Allowable Module Switch Current

The HP E1463A has an individual channel current specification of 5 A. However, if you apply the 5 A to all the channels with a relay contact resistance of .25 ohms, the power dissipation would be 200 W. Since the HP E1401A mainframe can only provide cooling for 60 W per slot (keeps the temperature rise to 10° C), this cannot be allowed to happen.

A reasonable maximum current for the entire module is 50 A. That is, 10 channels each carrying 5 A or some combination of channels and currents that total 50 A. This will produce about 67.5 W of internal dissipation, leading to a 15° C temperature rise. Figure 1-11 shows how to derate the channels, in terms of current throughout the channels, to keep internal power dissipation under 45 W and 67.5 W or 10° C and 15° C temperature rise, respectively.

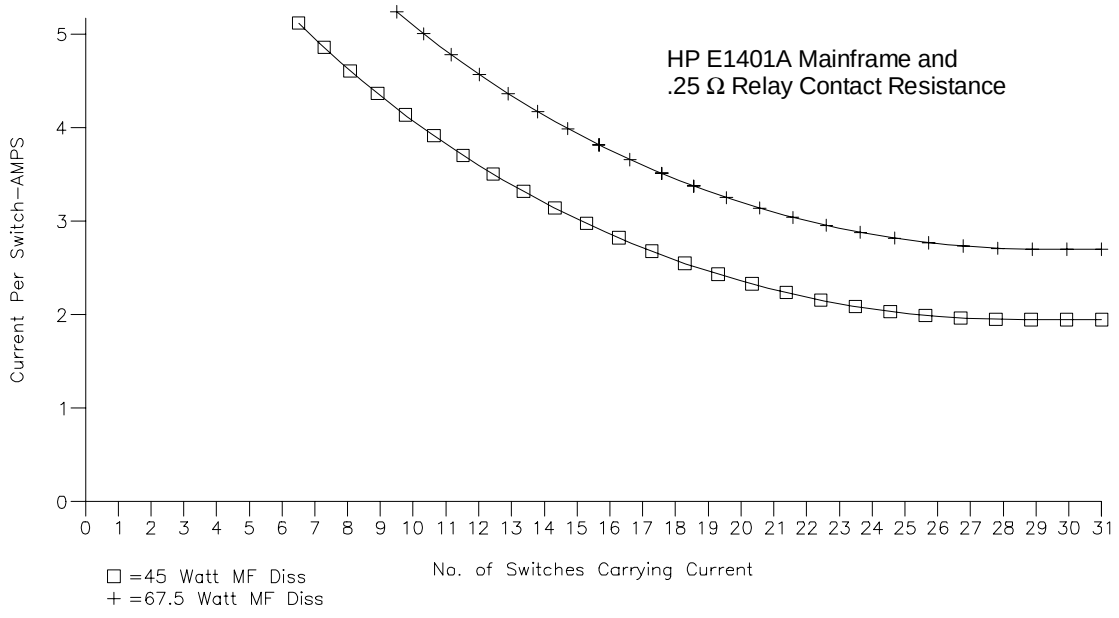


Figure 1-11. HP E1463A Allowable Switch Current

Programming the Form C Switch

There are several ways you can program the Form C switch module. One way is to write directly to the registers. This method can provide better throughput speed, however, it requires more knowledge of the Form C switch design. See Appendix B for information on register programming.

Another way to program the Form C switch module is to use an HP E1406 command module and SCPI commands. With SCPI commands the HP command module parses the commands and writes to the appropriate Form C switch register. The examples in this manual use the SCPI programming language. See Appendix B for examples on writing directly to the registers.

You can use different controllers and different programming languages. The examples in this manual, however, are based on the following configurations:

- HP 9000 Series 200/300 Computer running HP BASIC
- HP Vectra Computer (or compatible) with an HP 82335A HP-IB Interface Card (with command library) running Borland® Turbo C

See the *C-Size VXIbus System Installation and Getting Started Guide* for information on additional configurations.

Note Most examples in this manual use SCPI commands. See Appendix B for information on writing directly to the registers.

Specifying SCPI Commands

To address specific channels (relays) within a Form C switch module, you must specify the SCPI command and switch channel list. Table 1-3 lists the most commonly used commands.

Table 1-3. Common SCPI Commands

SCPI Command	Command Description
CLOSe <*channel_list*>	Connects the normally open (NO) terminal to the common (C) terminal for the specified channels.
OPEN <*channel_list*>	Connects the normally closed (NC) terminal to the common (C) terminal for the channels specified.
SCAN <*channel_list*>	Closes the set of Form C relays, one at a time.

Channel List

The *channel list* is a combination of the switch card number and the channel numbers. The *channel_list* takes the form of **@ccnn** where,

cc = switch card number (01-99)
nn = channel number (00-31)

Card Numbers

The card number (**cc** of the *channel list*) identifies the module within a switchbox. The card number assigned depends on the switch configuration used. Leading zeroes can be ignored for the card number.

Single-module Switchbox. In a single-module switchbox configuration, the card number is always 01.

Multiple-module Switchbox. In a multiple-module switchbox configuration, modules are set to successive logical addresses. The module with the lowest logical address is always card number 01. The module with the next successive logical address is card number 02, and so on. Figure 1-12 illustrates the card numbers and logical addresses of a typical multiple-module switchbox configuration. See the *C-Size VXIbus System Installation and Getting Started Guide* for additional switchbox instrument information.

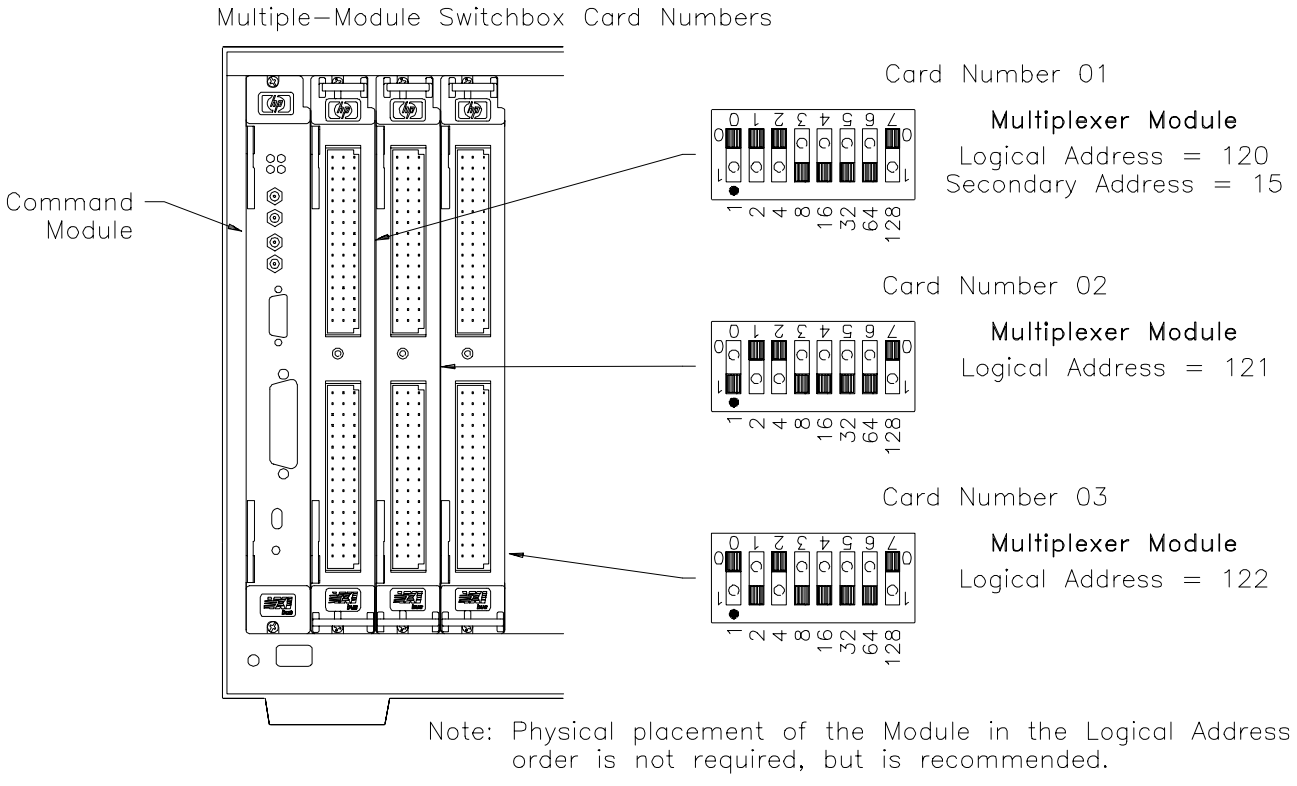


Figure 1-12. Card Numbers in a Multiple-module Switchbox

Channel Address

The channel address (**nn** of the *channel list*) determines which relay on the selected card will be addressed. Form C switch channel numbers are 00 through 31. The channels can be addressed using channel numbers or channel ranges. You can address the following:

- single channels (@ccnn);
- multiple channels (@ccnn,ccnn,...);
- sequential channels (@ccnn:ccnn);
- groups of sequential channels (@ccnn:ccnn,ccnn:ccnn);
- or any combination of the above.

Use a comma (,) to form a channel list or a colon (:) to form a channel range. Only valid channels can be accessed in a channel list or channel range. Also, the channel range must be from a lower channel number to a higher channel number. For example, CLOS (@100:215) is acceptable, but CLOS (@215:100) generates an error.

Initial Operation

Two example programs follow which use Hewlett-Packard BASIC and TURBO C languages to get you started using the Form C switch module. The first example assumes an HP 9000 Series 200/300 controller and a Hewlett-Packard Interface Bus (HP-IB). (HP-IB is the Hewlett-Packard implementation of the IEEE 488.2-1987 standard.) The second example assumes an HP Vectra Computer (or compatible) with an HP 82335A HP-IB Interface Card (with command library) running Borland® Turbo C.

This program closes channel 02 of a Form C switch module at logical address 120 (secondary address = 120/8 = 15) and queries the channel closure state. The result is returned to the computer and displayed (1 = channel closed, 0 = channel open). See Chapter 3 for information on the SCPI commands.

HP BASIC

```
10   !Reset the module.
20   OUTPUT  70915;"*RST"
30   !Close channel 02.
40   OUTPUT  70915;"CLOS  (@102)"
50   !Query channel 02 state.
60   OUTPUT  70915;"CLOS?  (@102)"
70   !Enter result into Value.
80   ENTER  70915;Value
90   !Display result (should print a "1" to indicate that the channel is closed).
100  PRINT Value
110  END
```

TURBO C

```
#include <stdio.h>
#include <chpib.h>                          /*Include file for HP-IB*/

#define ISC 7L
#define FORMC 70915L                        /*Form C default address*/
#define TASK1 "*RST"                        /*Command for a reset*/
#define TASK2 "CLOSE (@102)"                /*Command to close channel 02*/
#define TASK3 "CLOS? (@102)"                /*Command to query channel 02*/

main( )
 {
      char into[257];
       int length = 256;
                                            /*Output commands to Form C*/

      error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");

      error_handler (IOOUTPUTS (FORMC, TASK1, 4), "OUTPUT command");
      error_handler (IOOUTPUTS (FORMC, TASK2, 12), "OUTPUT command");
      error_handler (IOOUTPUTS (FORMC, TASK3, 12), "OUTPUT command");
                                            /*Enter from Form C*/

      error_handler (IOENTERS (FORMC, into, &length), "ENTER command");

      printf("Now let's see if the switch is closed: %s",into);

      return;
}
int error_handler (int error, char *routine)
{
      char ch;
      if (error != NOERR)
      {
         printf ("\n Error %d %s \n", error, errstr(error));
         printf (" in call to HP-IB function %s \n\n", routine);
         printf ("Press 'Enter' to exit: ");
         scanf ("%c", &ch);
         exit(0);
      }
      return 0;
}
```

Chapter 2
Using the HP E1463A Form C Switch

Using This Chapter

This chapter uses typical examples to show how to use the Form C switch module for switching channels and scanning channels. See Chapter 3 for command information. Chapter contents are as follows:

All examples in this chapter use the HP-IB select code of 7, primary address of 09, and a secondary address of 15 (LADDR=120).

Form C Switch Commands

Table 2-1 explains some of the SCPI commands used in this chapter. Refer to Chapter 3 for more information on these commands.

Table 2-1. Form C Switch Commands Used in Chapter 2

SCPI Command	Command Description
[ROUTe:]CLOSe <*channel_list*>	Closes the channels in the *channel list*.
[ROUTe:]CLOSe? <*channel_list*>	Queries the state of the channels in the *channel list*.
[ROUTe:]OPEN <*channel_list*>	Opens the channels in the *channel list*.
[ROUTe:]OPEN? <*channel_list*>	Queries the state of the channels in the *channel list*.
[ROUTe:]SCAN <*channel_list*>	Closes the channels in the *channel list*, one at a time.
INITiate[:IMMediate]	Starts the scan sequence and closes the first channel in the *channel list*.
TRIGger:SOURce BUS | EXT | HOLD | IMM | TTLT	Selects the trigger source to advance the scan.

Power-on and Reset Conditions

Since the Form C switch module has nonlatching relays, all relays are in the normally closed (NC) position at power-down and power-up.

The *RST command opens all channels, invalidates the current channel list for scanning, and sets the following:

Table 2-2. Reset Conditions

Parameter	Default	Description
ARM:COUNt	1	Number of scanning cycles is 1.
TRIGger:SOURce	IMM	Will advance scanning cycles automatically.
INITiate:CONTinuous	OFF	Number of scanning cycles set by ARM:COUNt.
OUTPut[:STATe]	OFF	Trigger output from EXT or TTL sources is disabled.

Module Identification

The following short programs use the *RST, *CLS, *IDN?, SYST:CTYP?, and SYST:CDES? commands to reset and identify the Form C switch module.

HP BASIC

```
10  !Dimensions three string variables to fifty characters.
20  DIM A$[50], B$[50], C$[50]
30  !Outputs the commands to reset and clear the status register.
40  OUTPUT  70915;  "*RST; *CLS"
50  !Queries for module identification.
60  OUTPUT  70915; "*IDN?"
70  !Enters the results into A$.
80  ENTER  70915; A$
90  !Output the command for a card description.
100 OUTPUT 70915; "SYST:CDES?  1"
110 !Enters the results into B$.
120 ENTER  70915; B$
130 !Outputs the command for the card type.
140 OUTPUT 70915; "SYST:CTYP? 1"
150 !Enters the results into C$.
160 ENTER  70915; C$
170 !Prints the contents of variables A$, B$, and C$.
180 PRINT A$, B$, C$
190 END
```

TURBO C

```
#include <stdio.h>
#include <chpib.h>                    /*Include file for HP-IB*/

#define ISC 7L
#define FORMC 70915L                  /*Form C default address*/
#define TASK1 "*RST;*CLS;*IDN?"       /*Reset, clear, and query identification*/
#define TASK2 "SYST:CDES? 1"          /*Command for card description*/
#define TASK3 "SYST:CTYP? 1"          /*Command for card type*/

main( )
{
     char into1[51], into2[51], into3[51];
     int length = 50;                 /*Output and enter commands to Form C*/

     error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");

     error_handler (IOOUTPUTS (FORMC, TASK1, 15), "OUTPUT command");
     error_handler (IOENTERS (FORMC, into1, &length), "ENTER command");

     error_handler (IOOUTPUTS (FORMC, TASK2, 12), "OUTPUT command");
     error_handler (IOENTERS (FORMC, into2, &length), "ENTER command");

     error_handler (IOOUTPUTS (FORMC, TASK3, 12), "OUTPUT command");
     error_handler (IOENTERS (FORMC, into3, &length), "ENTER command");

     printf("IDENTIFICATION: %s",into1);
     printf("CARD DESCRIPTION: %s",into2);
     printf("CARD TYPE: %s",into3);

     return;
}
int error_handler (int error, char *routine)
{
     char ch;
     if (error != NOERR)
     {
          printf ("\n Error %d %s \n", error, errstr(error));
          printf (" in call to HP-IB function %s \n\n", routine);
          printf ("Press 'Enter' to exit: ");
          scanf ("%c", &ch);
          exit(0);
     }
return 0;
}
```

A typical print for the HP E1463A will look like the following:

```
HEWLETT-PACKARD,SWITCHBOX,0,A.04.00
32 Channel General Purpose Relay
HEWLETT-PACKARD,E1463A,0,A.04.00
```

Switching Channels

For general purpose relay operation, you can connect or disconnect a load by opening or closing specified channel relays. By adding external pull-up resistors, the switch can be configured for digital output operations.

Use CLOS <*channel_list*> to connect a channel's normally open (NO) terminal to its common (C) terminal, or use OPEN <*channel_list*> to connect a channel's normally closed (NC) contact to its common (C) terminal. The *channel_list* has the form **(@ccnn)** where,

cc = card number (01-99)
nn = channel number (00-31)

To OPEN or CLOSe multiple channels, place a comma (,) between the channel numbers. For example, to close channels 101 and 103, execute CLOS (@101,103). To OPEN or CLOSe a continuous range of channels place a colon (:) between the first and last channel numbers.

The following HP BASIC program shows how to close and open channel 2 on an HP E1463A Form C module (card #1):

```
10  DISP  "TEST E1463A Module"
20  OUTPUT  70915; "ROUT:CLOS  (@102)"
30  OUTPUT  70915; "ROUT:OPEN  (@102)"
40  END
```

Note Implied commands are those which appear in square brackets (**[ ]**) in the command syntax. Note that the brackets are not part of the command and are not sent to the instrument. For example, as in the program above, ROUTe can be eliminated and just the CLOSe command can be used.

Example: Voltage Switching

This example closes channel 00 of a Form C switch module to switch the load voltage (E) from load 1 to load 2. When the channel relay is open, the load voltage is applied to load 1. When the relay is closed, the voltage is applied to load 2. See Figure 2-1 for typical user connections.

The following program shows how to close channel 00 of the HP E1463A Form C Switch:

```
10  DISP  "Testing the HP E1463A"
20  OUTPUT  70915; "CLOS  (@100)"   !Close channel 00 relay (connect
                                    NO to C). 1 is the card number
                                    and 00 is the channel number.
30  END
```

To open channel 00, use OPEN (@100).

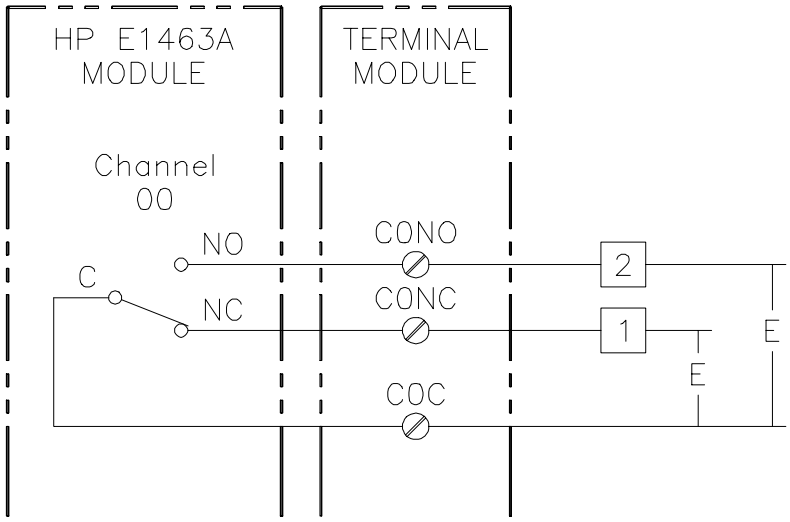


Figure 2-1. Voltage Switching

Example: Controlling RF Switches/ Step Attenuators

Figure 2-2 shows one way to drive the HP 8761 SPDT RF Switches or HP 33300 Series Programmable Step Attenuators. (This figure only shows control for the HP 33300 40 dB step. Additional drive relays are required for the 10 dB and 20 dB steps.) The HP 8761A and HP 33300A/C operate from a 12 - 15 V coil voltage, while the HP 8761B and HP 33300B/D operate from a 24 V - 30 V coil voltage. To close channel 00, execute:

```
10  DISP  "Applying -12V"
20  OUTPUT 70915; "CLOS  (@100)"    !Close channel 00 relay (connect
                                    NO to C). 1 is the card number
                                    and 00 is the channel number.
30  END
```

To open channel 00, use OPEN (@100).

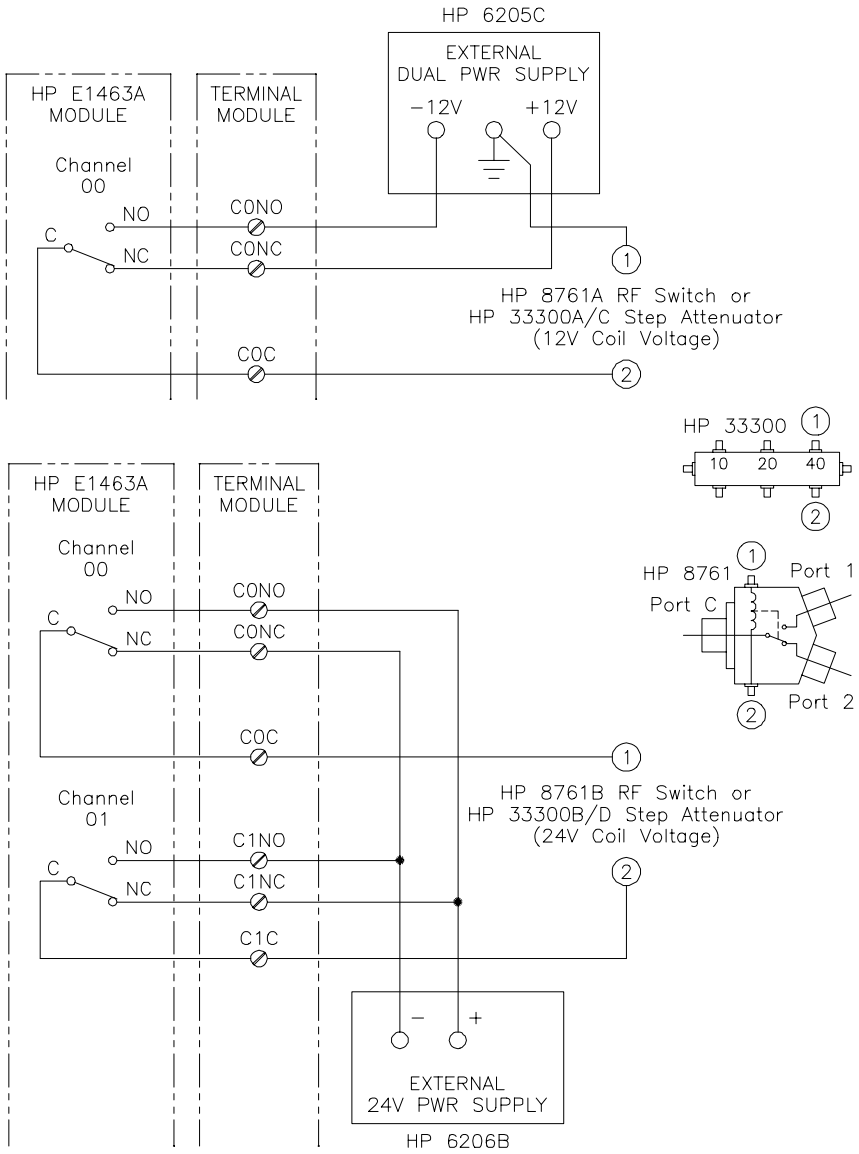


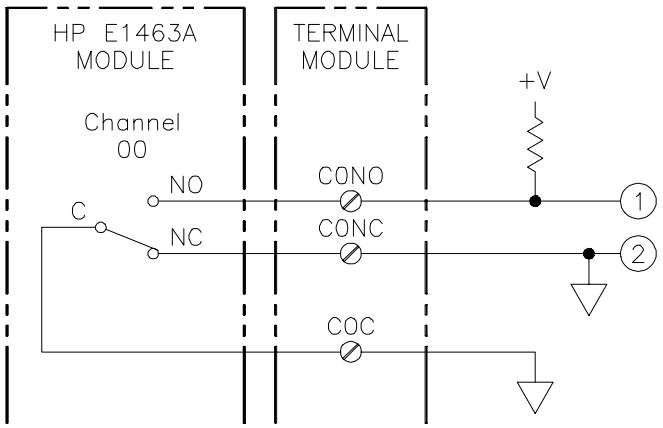
Figure 2-2. Controlling RF Switches/Step Attenuators

Example: Digital Output Configuration

Figure 2-3 shows channel 00 configured for digital output operation. When the channel 00 relay is open (NC connected to C), point 1 is at + V and point 2 is at 0 V. When the channel 00 relay is closed (NO connected to C), points 1 and 2 are both at 0 V. To close channel 00, execute the following:

```
10  DISP "Closing channel 0"
20  OUTPUT 70915; "CLOS  (@100)"   !Close channel 00 relay (connect
                                   NO to C). 1 is the card number
                                   and 00 is the channel number.
30  END
```

To open channel 00, use OPEN (@100).



Relay	1	2
Open	+V	0V
Closed	0V	0V

Figure 2-3. Digital Output Configuration

Example: Matrix Switching

The Form C switch module can be configured as a 4 x 8 single-wire matrix to connect any combination of up to four user sources (S0, S1, S2, S3) to any combination of up to eight user instruments (I0, I1, I1…I7) at a time. To do this you must make the following connections:

Connect Common (C) Channel Numbers Together	Connect Normally Open (NO) Channel Numbers Together
0, 8, 16, and 24	0 - 7
1, 9, 17, and 25	8 - 15
2, 10, 18, and 26	16 - 23
3, 11, 19, and 27	24 - 31
4, 12, 20, and 28	
5, 13, 21, and 29	
6, 14, 22, and 30	
7, 15, 23, and 31	

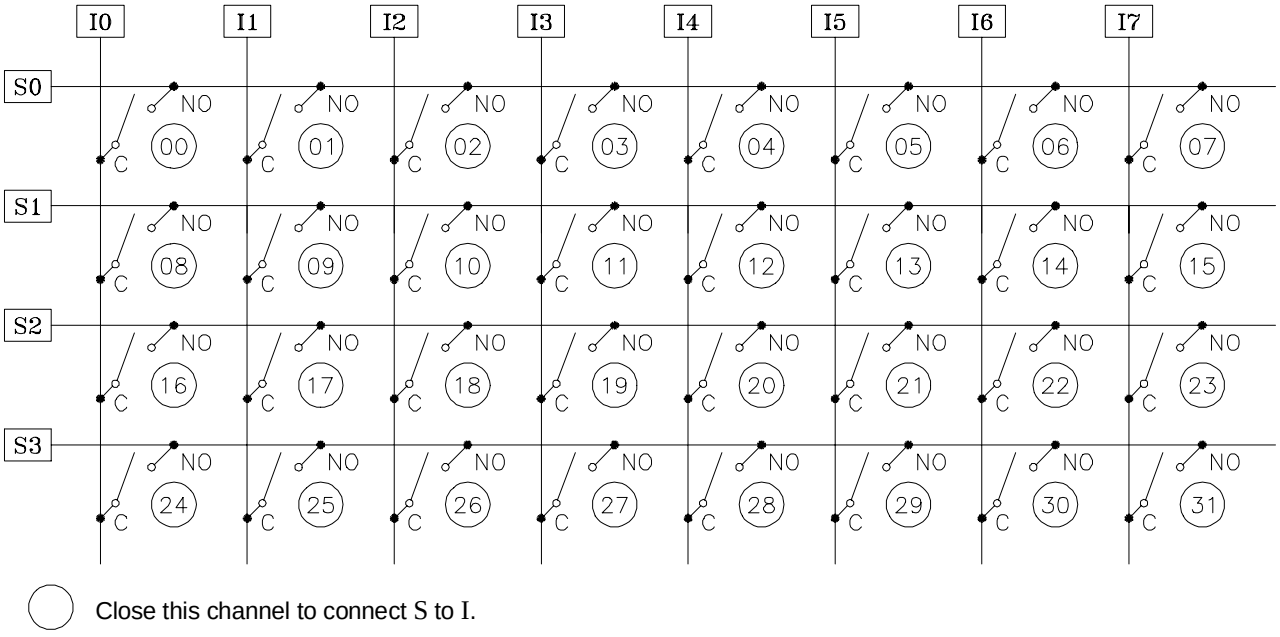


Figure 2-4. Matrix Switching

When the connections are completed you will have the following matrix:

Close the channel number enclosed in the circle to connect the corresponding row and column. The following example closes channel 25 to connect S3 to I1 and closes channel 20 to connect S2 to I4. To close channels 20 and 25, execute the following:

```
10  DISP "Testing Switch Matrix"
20  OUTPUT  70915; "CLOS  (@120,125)"!Close channels 20 and 25.
                                     1 is the card number; 20 and 25
                                     are channel numbers.
30  END
```

To open the channels, use OPEN (@120,125).

Scanning Channels

For the Form C switch module, scanning channels consists of closing a specified set of channels, one channel at a time. You can scan any combination of channels for a single-module or a multiple-module switchbox. Single, multiple, or continuous scanning modes are available. See Chapter 3 for additional information on scanning Form C switch channels.

Channel lists can extend across boundaries. For multiple-module switchbox instruments, the channels to be scanned can extend across switch modules. For example, for a two-module switchbox instrument, SCAN (@100:231) will scan all channels of both Form C switch modules.

Use ARM:COUNt <*number*> to set multiple/continuous scans (from 1 to 32,767 scans). Use INITiate:CONTinuous ON to set continuous scanning. See Chapter 3 for information about these SCPI commands.

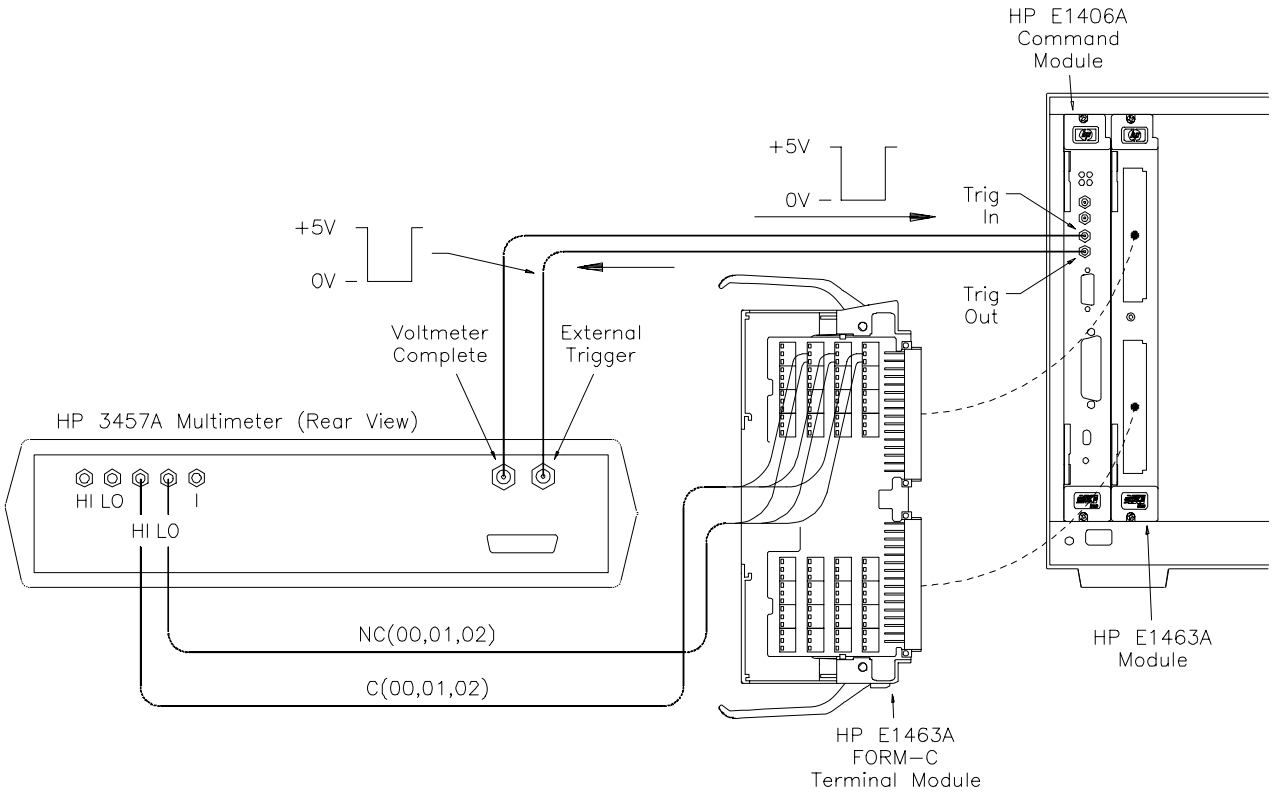


Figure 2-5. Scanning Using "Trig Out" Port

Example: Scanning Channels with an External Instrument Using “Trig In” and “Trig Out” Ports

This example shows one way to synchronize instrument measurements of a device under test (DUT) with Form C switch channel closures. For measurement synchronization, the HP E1406A "Trig In" and "Trig Out" ports are connected to the instrument "Voltmeter Complete" and "External Trigger" ports. See Figure 2-5 for typical user connections.

For this example, the normally closed (NC) contacts (channels 00-02) are connected to ground, and the measurements are made on the common (C) contacts. The command module and instrument are connected via HP-IB. The Form C switch module has a logical address 120 (secondary address 15), and the external instrument has an address of 722.

HP BASIC

```
10   !Reset and clear the module.
20   OUTPUT 70915; "*RST;*CLS"
30   !External trigger, dc volts.
40   OUTPUT 722;"TRIG EXT;DCV"
50   !Memory first in, first out.
60   OUTPUT 722;"MEM FIFO"
70   !Enable "Trig Out".
80   OUTPUT 70915;"OUTP ON"
90   !External triggering.
100  OUTPUT 70915;"TRIG:SOUR EXT"
110  !Scan channels 00-02.
120  OUTPUT 70915;"SCAN (@100:102)"
130  !Enable scan.
140  OUTPUT 70915;"INIT"
150  !Wait for switch closures.
160  WAIT 2
170  !Start loop.
180  FOR Channel=1 TO 3
190  !Enter result.
200  ENTER 722;Result
210  !Display result.
220  PRINT Result
230  !Increment count.
240  NEXT Channel
250  END
```

Example: Scanning Channels with System Multimeter Using TTL Trigger

This example uses the HP E1406A command module's TTL trigger bus lines to synchronize Form C channel closures to a system multimeter (HP E1412A). For measurement synchronization:

- HP E1406A TTL trigger bus line 0 is used by the Form C module to trigger the multimeter to perform a measurement.
- HP E1406A TTL trigger bus line 1 is used by the multimeter to advance the Form C scan.

These trigger bus lines are not actual hardware connections. The triggering is accomplished by the HP E1406A's firmware. The measurement is taken from the common (C) terminal. The common terminals for channels 0 through 2 are connected together for this example. When one of these switches is closed (C connected to NO), different DUTS are switched in for a measurement. Figure 2-6 shows how to connect the Form C module to the HP E1412A multimeter module. The connections shown with dotted lines are not actual hardware connections. These connections indicate how the firmware operates to accomplish the triggering.

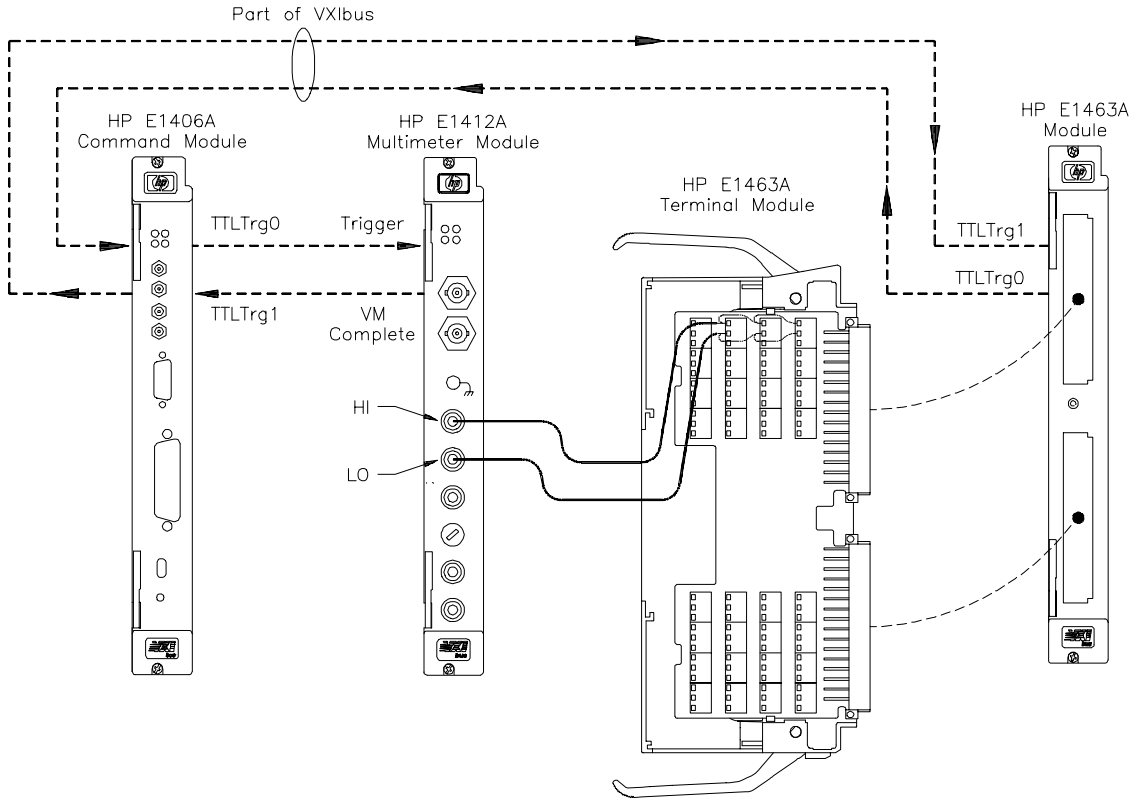


Figure 2-6. Scanning Using TTL Trigger Bus Lines

The following HP BASIC program sets up the multimeter (HP-IB address 70903) to scan making 2-wire resistance measurements.

HP BASIC

```
10   ALLOCATE REAL Rdgs(1:3)
20   !Reset and clear the modules.
30   OUTPUT  70915; "*RST;*CLS"
40   OUTPUT  70903; "*RST;*CLS"
50   !Multimeter triggers on TTL trigger line 0, multimeter pulses TTL trigger
60   !line 1 on measurement complete. Set multimeter function to resistance,
70   !range, NPLC.
80   OUTPUT  70903;"ABORT;:TRIG:SOUR  TTLTRG0"
90   OUTPUT  70903; "OUTP:TTLT1:STAT  ON"
100  OUTPUT  70903; "CONF:RES  AUTO,DEF"
110  OUTPUT  70903; "TRIG:DEL 0; COUN 3;:CAL:ZERO:AUTO ON"
120  !Check to see if multimeter ready; when ready, initialize trigger 1.
130  OUTPUT  70903; "*OPC?"
140  ENTER  70903; Check
150  OUTPUT  70903; "INIT"
160  !Set up the Form C: Form C pulses TTL Trigger line 0 on channel closed.
170  OUTPUT  70915; "OUTPUT:TTLT0:STATE ON"
180  !Set Form C to be triggered by TTL Trigger line 1.
190  OUTPUT  70915;"TRIG:SOUR  TTLT1"
200  OUTPUT  70915; "SCAN  (@100:102)"
210  OUTPUT  70915; "INIT"
220  !Enter and print readings.
230  OUTPUT  70903; "FETCH?"
240  ENTER  70903; Rdgs(*)
250  PRINT Rdgs(*)
260  END
```

Querying the Form C Switch Module

All query commands end with a "?". These commands are used to determine a specific state of the module. The data is sent to the output buffer where you can retrieve it into your computer. See Chapter 3 for more information on these commands.

Use CLOSe? <*channel_list*> or OPEN? <*channel_list*> to query the channel state (open/closed). CLOS? returns a "1" for channel(s) closed and a "0" for channel(s) open. OPEN? returns a "0" for channel(s) closed and a "1" for channel(s) open. (Commands are software queries and do not account for relay hardware failures.)

The following example closes a range of channels and queries for the results.

HP BASIC

```
10  !Dimensions a string variable to 32 characters.
20  DIM Channels$[32]
30  !Closes channels 00 through 31.
40  OUTPUT  70915;"CLOS  (@100:131)"
50  !Queries to see if the channels are closed.
60  OUTPUT  70915;"CLOS?  (@100:131)"
70  !Enters the results from the switch card into the variable Channels$.
80  ENTER  70915; Channels$
90  !Prints the channels closed (should print 1's).
100 PRINT "Channels Closed:";Channels$
110 END
```

Using the Scan Complete Bit

You can use the Scan Complete bit (bit 8) in the Operation Status Register (in the command module) of a switchbox to determine when a scanning cycle completes (no other bits in the register apply to the switchbox). Bit 8 has a decimal value of 256 and you can read it directly with the STAT:OPER? command. Refer to the STATus:OPERation[:EVENt]? command in Chapter 3 for an example.

When enabled by the STAT:OPER:ENAB 256 command, the Scan Complete bit will be reported as bit 7 of the Status Register. Use the HP-IB Serial Poll or the IEEE 488.2 Common Command *STB? to read the Status Register.

When bit 7 of the Status Register is enabled by the *SRE 128 Common Command to assert an HP-IB Service Request (SRQ), you can interrupt the computer when the Scan Complete bit is set, after a scanning cycle completes. This allows the computer to do other operations while the scanning cycle is in progress.

The following example monitors bit 7 in the Status Register to determine when the scanning cycle is complete. The computer used in this example is an HP 9000 Series 200/300 running HP BASIC as the programming language. The computer interfaces with an HP E1406A command module over HP-IB. The HP-IB select code is 7, the HP-IB primary address is 09, and the HP-IB secondary address is 15.

HP BASIC

```
10   !Resets and Clears the module.
20   OUTPUT  70915;"*RST; *CLS"
30   !Enable Scan Complete Bit.
40   OUTPUT  70915;"STAT:OPER:ENAB 256"
50   !Set the Form C switch up for continuous triggering.
60   OUTPUT  70915; "TRIG:SOUR IMM"
70   !Select channels to scan.
80   OUTPUT  70915; "SCAN (@100:115)"
90   !Wait for operation complete.
100   OUTPUT  70915; "*OPC?"
110  ENTER  70915; A$
120  PRINT "*OPC? = ";A$
130  !Query the contents in the operation status register.
140  OUTPUT  70915;"STAT:OPER:ENAB?"
150  ENTER  70915; A$
160  !Print the contents of the operation status register.
170  PRINT "STAT:OPER:ENAB?=";A$
180  !Query the contents of the status byte register.
190  OUTPUT  70915; "*STB?"
200  ENTER  70915; A$
210  !Print the contents of the status byte register.
220  PRINT "Switch Status = ";A$
230  !Start scan cycle.
240  OUTPUT  70915; "INIT"
250  !Initialize the value of the counter.
260  I = 0
270  !Stay in loop until some value is returned from the SPOLL (70915) command.
280  WHILE (I=0)
290     I = SPOLL(70915)
300     PRINT "Waiting for scan to complete: SPOLL = ";I
310  END WHILE
320  I = SPOLL(70915)
330  PRINT "Scan complete: spoll = ";I
340  END
```

Recalling and Saving States

The *SAV <*numeric_state*> command saves the current instrument state. The state number (0-9) is specified by the *numeric_state* parameter. The settings saved by this command are as follows:

- Channel relay states (open or closed)
- ARM:COUNt
- TRIGger:SOURce
- OUTPut:STATe
- INITiate:CONTinuous

The *RCL <*numeric_state*> command recalls the state when the last *SAV was executed for the specified *numeric_state* parameter (0-9). If no *SAV was executed for the *numeric_state*, *RST default settings are used. Refer to the *SAV settings list for the settings recalled by *RCL.

The following program shows how to save and recall Form C switch states.

HP BASIC

```
10   !Dimension a string variable for 150 characters.
20   DIM A$[150]
30   !Close channels 00 - 31 on the Form C.
40   OUTPUT  70915; "CLOS  (@100:131)"
50   !Save as numeric state 5.
60   OUTPUT  70915; "*SAV 5"
70   !Reset and clear the module.
80   OUTPUT  70915 "*RST;*CLS"
90   !Query the channels closed.
100  OUTPUT  70915;"CLOS?  (@100:131)"
110  ENTER  70915;A$
120  !Prints closed channels (should print 0's).
130  PRINT "Channels Closed:";A$
140  !Recall numeric state 5.
150  OUTPUT  70915; "*RCL 5"
160  !Query to see what channels are closed.
170  OUTPUT  70915 "CLOS?  (100:131)"
180  ENTER  70915;A$
190  !Print the closed channels (should print 1's).
200  PRINT "Channels Closed:";A$
210  END
```

Detecting Error Conditions

The SYSTem:ERRor? query requests a value from instrument's error register. This register contains an integer in the range [-32,768 to 32,767]. The response takes the following form:

<*err_number*>,<*err_message*>

where, <*err_number*> is the value of the instrument's error, and <*err_message*> is a short description of the error.

The following programs attempt an illegal channel closure and polls for an error message:

HP BASIC

```
10  !Dimension a string variable for 256 characters.
20  DIM Err_num$[256]
30  !Try to close an illegal channel.
40  OUTPUT  70915; "CLOS  (@135)"
50  !Query for a system error.
60  OUTPUT  70915; "SYST:ERR?"
70  ENTER  70915; Err_num$
80  !Prints error +2001, "Invalid channel number".
90  PRINT Err_num$
100 END
```

TURBO C

```
#include <stdio.h>
#include <chpib.h>                  /*Include file for HP-IB*/

#define ISC 7L
#define FORMC 70915L                /*Form C default address*/
#define TASK1 "CLOSE (@135)"        /*Command for illegal switch closure*/
#define TASK2 "SYST:ERR?"           /*Command for system error*/

main( )
{
      char into[257];
      int length = 256;

                                    /*Output commands to Form C*/

      error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");

      error_handler (IOOUTPUTS (FORMC, TASK1, 12), "OUTPUT command");
      error_handler (IOOUTPUTS (FORMC, TASK2, 9), "OUTPUT command");

                                    /*Enter from Form C*/

      error_handler (IOENTERS (FORMC, into, &length), "ENTER command");

      printf("Now let's print the errors: %s",into);

      return;
}
int error_handler (int error, char *routine)
{
      char ch;
      if (error != NOERR)
      {
      printf ("\n Error %d %s \n", error, errstr(error));
      printf (" in call to HP-IB function %s \n\n", routine);
      printf ("Press 'Enter' to exit: ");
            scanf ("%c", &ch);
            exit(0);
      }
      return 0;
}
```

If no error occurs, the switchbox responds with `0, "No error"`. If there has been more than one error, the instrument will respond with the first one in its error queue. Subsequent queries continue to read the error queue until it is empty. The maximum <*err_message*> string length is 255 characters.

Synchronizing the Form C Switch

The following example shows how to synchronize a Form C switch module with a measurement instrument. In this example, the Form C switch module switches a signal to a multimeter. The program then verifies that the channel is closed before the multimeter begins its measurement.

HP BASIC

```
10   !Closes channel 5.
20   OUTPUT  70915; "CLOS  (@105)"
30   !Wait for operation complete.
40   OUTPUT  70915; "*OPC?"
50   ENTER  70915; Opc_value
60   !Check to see if channel is closed.
70   OUTPUT  70915; "CLOS?  (@105)"
80   ENTER  70915;A
90   !When channel is closed, measure the voltage.
100  If A=1 THEN
110     OUTPUT  70903;"MEAS:VOLT:DC?"
120     ENTER  70903; Meas_value
130  !Print the measured voltage.
140     PRINT  Meas_value
150  ELSE
160     PRINT  "CHANNEL DID NOT CLOSE"
170  END IF
180  END
```

Chapter 3
HP E1463A Form C Switch Command Reference

Using This Chapter

This chapter describes Standard Commands for Programmable Instruments (SCPI) commands and summarizes IEEE 488.2 Common (*) Commands used in this manual.

See the *HP E1406A Command Module User's Manual* for additional information on SCPI and common commands. Chapter contents are as follows:

Command Types

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

Common Command Format

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

```
*RST          *ESE <unmask>          *STB?
```

SCPI Command Format

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

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

[ROUTe:] is the root command, CLOSe and SCAN are the second level sub commands with <*channel_list*> as a parameter, and :MODE? is a third level command. [ROUTe:] is also an implied command and is, therefore, optional.

Note There must be a space between the second level command (CLOSe, for example) and the parameter <*channel_list*>.

Command Separator

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

```
[ROUTe:]SCAN:MODE?
```

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

Abbreviated Commands

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

For example, if the command syntax shows TRIGger, then TRIG and TRIGGER are both acceptable forms. Other forms of TRIGger, such as TRIGG or TRIGGE will generate an error. You may use upper or lower case letters. Therefore, TRIGGER, trigger, and TrIgGeR are all acceptable.

Implied Commands

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

```
[ROUTe:]
    CLOSe? <channel_list>
```

The root command [ROUTe:] is an implied command. To make a query about a channel's present status, you can send either of the following command statements:

ROUT:CLOSe? <*channel_list*> ***or*** CLOSe? <*channel_list*>

Variable Command Syntax

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

OUTPut:TTLTrg*n*

In this command, the "*n*" is replaced by a number. No space is left between the command and the number because the number is not a parameter.

The number is part of the command syntax. In the case of OUTPut:TTLTrg*n*, "*n*" can range from 0 through 7.

Parameter Types

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

- **Boolean Parameters** represent a single binary condition that is either true or false (for example, ON, OFF, 1, 0). Any non-zero value is considered true.
- **Discrete Parameters** selects from a finite number of values. These parameters use mnemonics to represent each valid setting. An example is the TRIGger:SOURce <*source*> command where *source* can be BUS, EXTernal, HOLD, IMMediate, or TTLTrg*n*.
- **Numeric Parameters** are commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation (for example, 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E- 01). Special cases include MINimum, MAXimum, DEFault, and INFinity.
- **Optional Parameters** are shown within square brackets (**[ ]**). The brackets are not part of the command, and are not sent to the instrument. If you do not specify a value for an optional parameter, the instrument chooses a default value. For example, consider the ARM:COUNt? [<MIN | MAX>] command. If you send the command without specifying a parameter, the present ARM:COUNt value is returned. If you send the MIN parameter, the command returns the minimum count available. If you send the MAX parameter, the command returns the maximum count available. Be sure to place a space between the command and the parameter.

Linking Commands

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

```
*RST;*RCL 1      or      CLOS (@101);*SAV 1
```

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

```
CLOS (@101);:CLOS? (@101)
```

SCPI also allows several commands within the same subsystem to be linked with a semicolon. For example:

```
ROUT:CLOS (@101);:ROUT:CLOS? (@101)
```

or

```
ROUT:CLOS (@101);CLOS? (@101)
```

SCPI Command Reference

This section describes the Standard Commands for Programmable Instruments (SCPI) reference commands for the Form C switch. Commands are listed alphabetically by subsystem and also within each subsystem.

ABORt

The ABORt command stops a scan in progress when the scan is enabled via the interface and the trigger source is TRIGger:SOURce BUS or TRIGger:SOURce HOLD.

Subsystem Syntax

```
ABORt
```

Comments

- **ABORt Actions:** The ABORt command terminates the scan and invalidates the current channel list.
- **Stopping Scan Enabled Via Interface:** When a scan is enabled via an interface, an interface CLEAR command can be used to stop the scan. When the scan is enabled via the interface and TRIG:SOUR BUS or HOLD is set, you can use ABORt to stop the scan.
- **Related Commands:** ARM, INITiate:CONTinuous, [ROUTe:]SCAN, TRIGger

Example

Stopping a Scan with ABORt

This example stops a continuous scan in progress.

```
TRIG:SOUR  BUS          !Trigger command will be via
                        backplane (bus) interface
                        (*TRG command generates trigger).
INIT:CONT  ON           !Set continuous scanning.
SCAN  (@100:107)        !Scan channels 00 to 07.
INIT                    !Start scan, close channel 00.
   .
   .
   .
ABOR                    !Abort scan in progress.
```

ARM

The ARM subsystem selects the number of scanning cycles (1 to 32,767) for each INITiate command.

Subsystem Syntax

```
ARM
    :COUNt <number> MIN | MAX
    :COUNt? [<MIN | MAX>]
```

:COUNt

ARM:COUNt <*number*> MIN | MAX allows scanning cycles to occur a multiple of times (1 to 32,767) with one INITiate command when INITiate:CONTinuous OFF | 0 is set. MIN sets 1 cycle and MAX sets 32,767 cycles.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
number	numeric	1 - 32,767 | MIN | MAX	1

Comments

- **Number of Scans:** Use only numeric values between 1 and 32767, MIN, or MAX for the number of scanning cycles.
- **Related Commands:** ABORt, INITiate[:IMMediate]
- ***RST Condition:** ARM:COUNt 1

Example

Setting Ten Scanning Cycles

This example sets a Form C switch for 10 scans of channels 00 through 03. When the scan sequence completes, channels 00 through 03 (relays 00 through 03) are closed.

```
ARM:COUN  10          !Set 10 scans per INIT command.
SCAN  (@100:103)      !Scan channels 00 to 03.
INIT                  !Start scan, close channel 00.
```

:COUNt?

ARM:COUNt? [<MIN | MAX>] returns the current number of scanning cycles set by ARM:COUNt. The current number of scan cycles is returned when MIN or MAX is not specified. With MIN or MAX as a parameter, MIN returns "1" and MAX returns "32,767".

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
MIN | MAX	numeric	MIN = 1, MAX = 32,767	current cycles

Comments

- **Related Commands:** INITiate[:IMMediate]

Example

Query Number of Scans

This example sets a switchbox for 10 scanning cycles and queries the number of scan cycles set. The ARM:COUN? command returns 10.

```
ARM:COUN  10          !Set 10 scans per INIT command.
ARM:COUN?             !Query number of scans.
```

DISPlay

The DISPlay subsystem monitors the channel state of the selected module in a switchbox. This subsystem operates with an HP E1406A command module when a display terminal is connected.

Subsystem Syntax

```
DISPlay
    :MONitor
        :CARD <number> | AUTO
        [:STATe] <mode>
```

:MONitor:CARD

DISPlay:MONitor:CARD <*number*> | AUTO selects the module in a switchbox to be monitored.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
number | AUTO	numeric	1 - 99	AUTO

Comments

- **Selecting a Specific Module to be Monitored:** Use the DISPlay:MONitor:CARD command to send the card number for the switchbox to be monitored.
- **Selecting the Present Module to be Monitored:** Use the DISPlay:MONitor:CARD AUTO command to select the last module addressed by a switching command (for example, [ROUTe:]CLOSe).
- ***RST Conditions:** DISPlay:MONitor:CARD AUTO

Example

Select Module #2 in a Switchbox for Monitoring

```
DISP:MON:CARD 2          !Selects module #2 in a switchbox.
```

:MONitor[:STATe]

DISPlay:MONitor[:STATe] <*mode*> turns the monitor mode ON or OFF.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
mode	boolean	ON | OFF | 1 | 0	OFF | 0

Comments

- **Monitoring Switchbox Channels:** DISPlay:MONitor:STATe ON or DISPlay:MONitor:STATe 1 turns the monitor mode ON to show the channel state of the selected module.

  DISPlay:MONitor:STATe OFF or DISPlay:MONitor:STATe 0 turns the channel monitor OFF.

- **Selecting the Module to be Monitored:** Use the DISPlay:MONitor:CARD <*number*> AUTO command to select the module.

- **Monitor Mode with an HP E1463A:** When monitoring mode is turned ON, decimal numbers representing the channels closed will be displayed at the bottom of the display terminal. For example, if channels 3, 7, and 12 are closed, the bottom of the display will read as follows:

  ```
  Chan , , ,3, , , ,7, , , , ,12, , , , , . . . etc.
  ```

  The channel numbers represent channels that are closed.

- ***RST Condition:** DISPlay:MONitor[:STATe] OFF | 0

Example

Enabling the Monitor Mode

```
DISP:MON:CARD 2          !Select module #2 in a switchbox.
DISP:MON 1               !Turn monitor mode ON.
```

INITiate

The INITiate command subsystem selects continuous scanning cycles and starts the scanning cycle.

Subsystem Syntax

```
INITiate
    :CONTinuous <mode>
    :CONTinuous?
    [:IMMediate]
```

:CONTinuous

INITiate:CONTinuous *<mode>* enables or disables continuous scanning cycles for the switchbox.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
mode	boolean	0 | 1 | OFF | ON	OFF | 0

Comments

- **Continuous Scanning Operation:** Continuous scanning is enabled with the INITiate:CONTinuous ON or INITiate:CONTinuous 1 command. Sending the INITiate:IMMediate command closes the first channel in the channel list. Each trigger from the source specified by the TRIGger:SOURce command advances the scan through the channel list. A trigger at the end of the channel list closes the first channel in the channel list and the scan cycle repeats.
- **Noncontinuous Scanning Operation:** Noncontinuous scanning is enabled with the INITiate:CONTinuous OFF or INITiate:CONTinuous 0 command. Sending the INITiate:IMMediate command closes the first channel in the channel list. Each trigger from the source specified by the TRIGger:SOURce command advances the scan through the channel list. At the end of the scanning cycle, the last channel in the channel list is opened.
- **Stopping Continuous Scan:** See the ABORt command on page 52.
- **Related Commands:** ABORt, ARM:COUNt, TRIGger:SOURce
- ***RST Condition:** INITiate:CONTinuous OFF | 0

Example

Enabling Continuous Scanning

This example enables continuous scanning of channels 00 through 03 of a single-module switchbox. Since TRIGger:SOURce IMMediate (default) is set, use an interface clear command (such as CLEAR) to stop the scan.

```
INIT:CONT ON          !Enable continuous scanning.
SCAN (@100:103)       !Define channel list.
INIT                  !Start scan cycle, close channel 00.
```

:CONTinuous?

INITiate:CONTinuous? queries the scanning state. With continuous scanning enabled, the command returns "1" (ON). With continuous scanning disabled, the command returns "0" (OFF).

Example

Query Continuous Scanning State

This example enables continuous scanning of a switchbox and queries the state. Since continuous scanning is enabled, INIT:CONT? returns "1".

```
INIT:CONT  ON                    !Enable continuous scanning.
INIT:CONT?                       !Query continuous scanning state.
```

[:IMMediate]

INITiate[:IMMediate] starts the scanning process and closes the first channel in the channel list. Successive triggers from the source specified by the TRIGger:SOURce command advances the scan through the channel list.

Comments

- **Starting the Scanning Cycle:** The INITiate:IMMediate command starts scanning by closing the first channel in the channel list. Each trigger received advances the scan to the next channel in the channel list. An invalid channel list definition causes an error (see [ROUTe:]SCAN on page 65).
- **Stopping Scanning Cycles:** See the ABORt command on page 52.

Example

Enabling a Single Scan

This example enables a single scan of channels 00 through 03 of a single-module switchbox. The trigger source to advance the scan is immediate (internal) triggering set with TRIGger:SOURce IMMediate (default).

```
SCAN  (@100:103)                 !Scan channels 00 - 03.
INIT                             !Begin scan, close channel 00
                                 (use immediate triggering).
```

OUTPut

The OUTPut command subsystem enables or disables the different trigger lines of the HP E1406A command module.

Subsystem Syntax

```
OUTPut
    :EXTernal
        [:STATe] <mode>
        [:STATe]?
    [:STATe] <mode>
    [:STATe]?
    :TTLTrgn (:TTLTrg0 through :TTLTrg7)
        [:STATe] <mode>
        [:STATe]?
```

:EXTernal[:STATe]

OUTPut:EXTernal[:STATe] <*mode*> enables or disables the "Trig Out" port on the HP E1406A command module.

- OUTPut:EXTernal[:STATe] ON | 1 enables the port
- OUTPut:EXTernal[:STATe] OFF | 0 disables the port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
mode	boolean	ON | OFF | 1 | 0	OFF | 0

Comments

- **Enabling "Trig Out" Port:** When enabled, a pulse is output from the "Trig Out" port after each scanned switchbox channel is closed. If disabled, a pulse is not output from the port after channel closures. The output pulse is a + 5 V negative-going pulse.
- **"Trig Out" Port Shared by Switchboxes:** When enabled, the "Trig Out" port is pulsed by any switchbox each time a scanned channel is closed. To disable the output for a specific module send the OUTPut:EXTernal[:STATe] OFF or OUTPut:EXTernal[:STATe] 0 command for that module.
- **One Output Selected at a Time:** Only one output (TTLTrg or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output.
- **Related Commands:** [ROUTe:]SCAN, TRIGger:SOURce
- ***RST Condition:** OUTPut:EXTernal[:STATe] OFF (port disabled)

Example

Enabling "Trig Out" Port

```
OUTP:EXT  ON                    !Enable "Trig Out" port to output
                                pulse after each scanned channel is
                                closed.
```

:EXTernal[:STATe]?

OUTPut:EXTernal[:STATe]? queries the present state of the "Trig Out" port. The command returns "1" if the port is enabled or "0" if the port is disabled.

Example

Query "Trig Out" Port Enable State

This example enables the "Trig Out" port and queries the enable state. The OUTPut:EXTernal[:STATe]? command returns "1" since the port is enabled.

```
OUTP:EXT  ON                    !Enable "Trig Out" port.
OUTP:EXT?                       !Query port enable state.
```

[:STATe]

OUTPut[:STATe] <*mode*> enables or disables the "Trig Out" port on the HP E1406A command module. OUTPut[:STATe] ON | 1 enables the port and OUTPut[:STATe] OFF | 0 disables the port. This command functions the same as the OUTPut:EXTernal[:STATe] command.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
mode	boolean	0 | 1 | OFF | ON	OFF | 0

Comments

- ***RST Condition:** OUTPut[:STATe] OFF (port disabled)

Example

Enabling "Trig Out" Port

```
OUTP  ON                        !Enable "Trig Out" port to output
                                pulse after each scanned channel is
                                closed.
```

[:STATe]?

OUTPut[:STATe]? queries the present state of the "Trig Out" port. The command returns "1" if the port is enabled or "0" if the port is disabled. This command functions the same as the OUTPut:EXTernal[:STATe]? command.

Example

Query "Trig Out" Port Enable State

This example enables the "Trig Out" port and queries the enable state. The OUTPut[:STATe]? command returns "1" since the port is enabled.

```
OUTP  ON                        !Enable "Trig Out" port.
OUTP?                           !Query port enable state.
```

:TTLTrg*n*[:STATe]

OUTPut:TTLTrg*n*[:STATe] <*mode*> selects and enables which TTL Trigger bus line (0 to 7) will output a trigger when a channel is closed during a scan. This is also used to disable a selected TTL Trigger bus line. "*n*" specifies the TTL Trigger bus line (0 to 7) and "*mode*" enables (ON or 1) or disables (OFF or 0) the specified TTL Trigger bus line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
n	numeric	0 to 7	N/A
mode	boolean	0 | 1 | OFF | ON	OFF | 0

Comments

- **Enabling TTL Trigger Bus:** When enabled, a pulse is output from the selected TTL Trigger bus line (0 to 7) after each channel in the switchbox is closed during a scan. If disabled, a pulse is not output. The output is a negative-going pulse.
- **One Output Selected at a Time:** Only one output (TTLTrg or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if TTLTrg1 is the active output and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active output.
- **Related Commands:** [ROUTe:]SCAN, TRIGger:SOURce, OUTPut:TTLTrg*n*[:STATe]?
- ***RST Condition:** OUTPut:TTLTrg*n*[:STATe] OFF (disabled)

Example

Enabling TTL Trigger Bus Line 7

```
OUTP:TTLT7:STAT 1          !Enable TTL Trigger bus line 7 to
                           output pulse after each scanned
                           channel is closed.
```

:TTLTrg*n*[:STATe]?

OUTPut:TTLTrg*n*[:STATe]? queries the present state of the specified TTL Trigger bus line. The command returns "1" if the specified TTLTrg bus line is enabled or "0" if disabled.

Example

Query TTL Trigger Bus Enable State

This example enables TTL Trigger bus line 7 and queries the enable state. The OUTPut:TTLTrg*n*? command returns "1" since the port is enabled.

```
OUTP:TTLT7:STAT 1          !Enable TTL Trigger bus line 7.
OUTP:TTLT7?                !Query bus enable state.
```

[ROUTe:]

The [ROUTe:] command subsystem controls switching and scanning operations for Form C switch modules in a switchbox.

Subsystem Syntax

```
[ROUTe:]
    CLOSe <channel_list>
    CLOSe? <channel_list>
    OPEN <channel_list>
    OPEN? <channel_list>
    SCAN <channel_list>
```

Note There must be a space between the second level command (CLOS, for example) and the parameter <*channel_list*>.

CLOSe

[ROUTe:]CLOSe <*channel_list*> closes the Form C switch channels specified by *channel_list*. *Channel_list* has the form (@ccnn) where cc = card number (01-99) and nn = channel number (00-31).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
channel_list	numeric	cc00 - cc31	N/A

Comments

- **Closing Channels:** To close:
  - a single channel use ROUT:CLOS (@ccnn);
  - multiple channels use ROUT:CLOS (@ccnn,ccnn,...);
  - sequential channels use ROUT:CLOS (@ccnn:ccnn);
  - groups of sequential channels use ROUT:CLOS (@ccnn:ccnn,ccnn:ccnn);
  - or any combination of the above.

  Closure order for multiple channels with a single command is not guaranteed.

Note Channel numbers can be in the *channel_list* in any random order.

- **Related Commands:** [ROUTe:]OPEN, [ROUTe:]CLOSe?
- ***RST Condition:** All channels open.

Example Closing Form C Switch Channels

This example closes channels 100 and 213 of a two-module switchbox (card numbers 01 and 02).

```
CLOS  (@100,213)          !Close channels 100 and 213. 100
                          closes channel 00 of card #1 and
                          213 closes channel 13 of card #2.
```

CLOSe?

[ROUTe:]CLOSe? <*channel_list*> returns the current state of the channel(s) queried. *Channel_list* has the form (@ccnn) (see [ROUTe:]CLOSe on page 62 for definition). The command returns "1" if channel(s) are closed or returns "0" if channel(s) are open.

Comments

- **Query is Software Readback:** The ROUTe:CLOSe? command returns the current software state of the channel(s) specified. It does not account for relay hardware failures.

Note

A maximum of 128 channels can be queried at one time. Therefore, if you want to query more than 128 channels, you must enter the query data in two separate commands.

Example Query Channel Closure

This example closes channels 100 and 213 of a two-module switchbox and queries channel closure. Since the channels are programmed to be closed "1,1" is returned as a string.

```
CLOS  (@100,213)          !Close channels 100 and 213.
CLOS?  (@100,213)         !Query channels 100 and 213 state.
```

OPEN

[ROUTe:]OPEN <*channel_list*> opens the Form C switch channels specified by *channel_list*. *Channel_list* has the form (@ccnn) where cc = card number (01-99) and nn = channel number (00-31).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
channel_list	numeric	cc00 - cc31	N/A

Comments

- **Opening Channels:** To open:
  - a single channel use ROUT:OPEN (@ccnn);
  - multiple channels use ROUT:OPEN (@ccnn,ccnn,...);
  - sequential channels use ROUT:OPEN (@ccnn:ccnn);
  - groups of sequential channels use ROUT:OPEN (@ccnn:ccnn,ccnn:ccnn);
  - or any combination of the above.

  Opening order for multiple channels with a single command is not guaranteed.

- **Related Commands:** [ROUTe:]CLOSe, [ROUTe:]OPEN?

- ***RST Condition:** All channels open.

Example

Opening Form C Switch Channels

This example opens channels 100 and 213 of a two-module switchbox (card numbers 01 and 02).

```
OPEN (@100,213)          !Open channels 100 and 213. 100 opens channel 00 of card #1 and 213 opens channel 13 of card #2.
```

OPEN?

[ROUTe:]OPEN? <*channel_list*> returns the current state of the channel(s) queried. *Channel_list* has the form (@ccnn) (see [ROUTe:]OPEN on page 64 for definition). The command returns "1" if channel(s) are open or returns "0" if channel(s) are closed.

Comments

- **Query is Software Readback:** The ROUTe:OPEN? command returns the current software state of the channel(s) specified. It does not account for relay hardware failures.

Note

A maximum of 128 channels can be queried at one time. Therefore, if you want to query more than 128 channels, you must enter the query data in two separate commands.

Example Query Channel Open State

This example opens channels 100 and 213 of a two-module switchbox and queries channel 213 state. Since channel 213 is programmed to be open, "1" is returned.

```
OPEN  (@100,213)          !Open channels 100 and 213.
OPEN?  (@213)             !Query channel 213 state.
```

SCAN

[ROUTe:]SCAN <*channel_list*> defines the channels to be scanned. *Channel_list* has the form (@ccnn) where cc = card number (01-99) and nn = channel number (00-31).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
channel_list	numeric	cc00 - cc31	N/A

Comments

- **Defining Scan List:** When ROUTe:SCAN is executed, the channel list is checked for valid card and channel numbers. An error is generated for an invalid channel list.
- **Scanning Channels:** To scan:
  - a single channel use ROUT:SCAN (@ccnn);
  - multiple channels use ROUT:SCAN (@ccnn,ccnn,...);
  - sequential channels use ROUT:SCAN (@ccnn:ccnn);
  - groups of sequential channels use ROUT:SCAN (@ccnn:ccnn,ccnn:ccnn);
  - or any combination of the above.

Note Channel numbers can be in the *channel_list* in any random order.

- **Scanning Operation:** When a valid channel list is defined, INITiate[:IMMediate] begins the scan and closes the first channel in the *channel_list*. Successive triggers from the source specified by TRIGger:SOURce advance the scan through the *channel list*. At the end of the scan, the last trigger opens the last channel.
- **Stopping Scan:** See the ABORt command on page 52.
- **Related Commands:** TRIGger, TRIGger:SOURce
- ***RST Condition:** All channels open.

See Chapter 2 for example scanning programs using external instruments.

Example Scanning Using External Device

See "Scanning Channels" beginning on page 39 for examples of scanning programs using external instruments.

STATus

The STATus subsystem reports the bit values of the Operation Status Register. It also allows you to unmask the bits you want reported from the Standard Event Register and to read the summary bits from the Status Byte Register.

Subsystem Syntax

```
STATus
    :OPERation
        :CONDition?
        :ENABle <unmask>
        :ENABle?
        [:EVENt?]
    :PRESet
```

The STATus system contains four registers, two of which are under IEEE 488.2 control; the Standard Event Status Register (*ESE?) and the Status Byte Register (*STB?). The operational status bit (OPR), service request bit (RQS), standard event summary bit (ESB), message available bit (MAV) and questionable data bit (QUE) in the Status Byte Register (bits 7, 6, 5, 4 and 3 respectively) can be queried with the *STB? command. Use the *ESE? command to query the "*unmask*" value for the Standard Event Status Register (the bits you want logically OR'd into the summary bit). The registers are queried using decimal weighted bit values. The decimal equivalents for bits 0 through 15 are included in Figure 3-1.

A numeric value of 256 executed in a STAT:OPER:ENABle <*unmask*> command allows only bit 8 to generate a summary bit. The decimal value for bit 8 is 256.

The decimal values are also used in the inverse manner to determine which bits are set from the total value returned by an EVENt or CONDition query. The Form C switch driver exploits only bit 8 of Operation Status Register. This bit is called the scan complete bit which is set whenever a scan operation completes. Since completion of a scan operation is an event in time, you will find that bit 8 will never appear set when STAT:OPER:COND? is queried. However, you can find bit 8 set with the STAT:OPER:EVEN? query command.

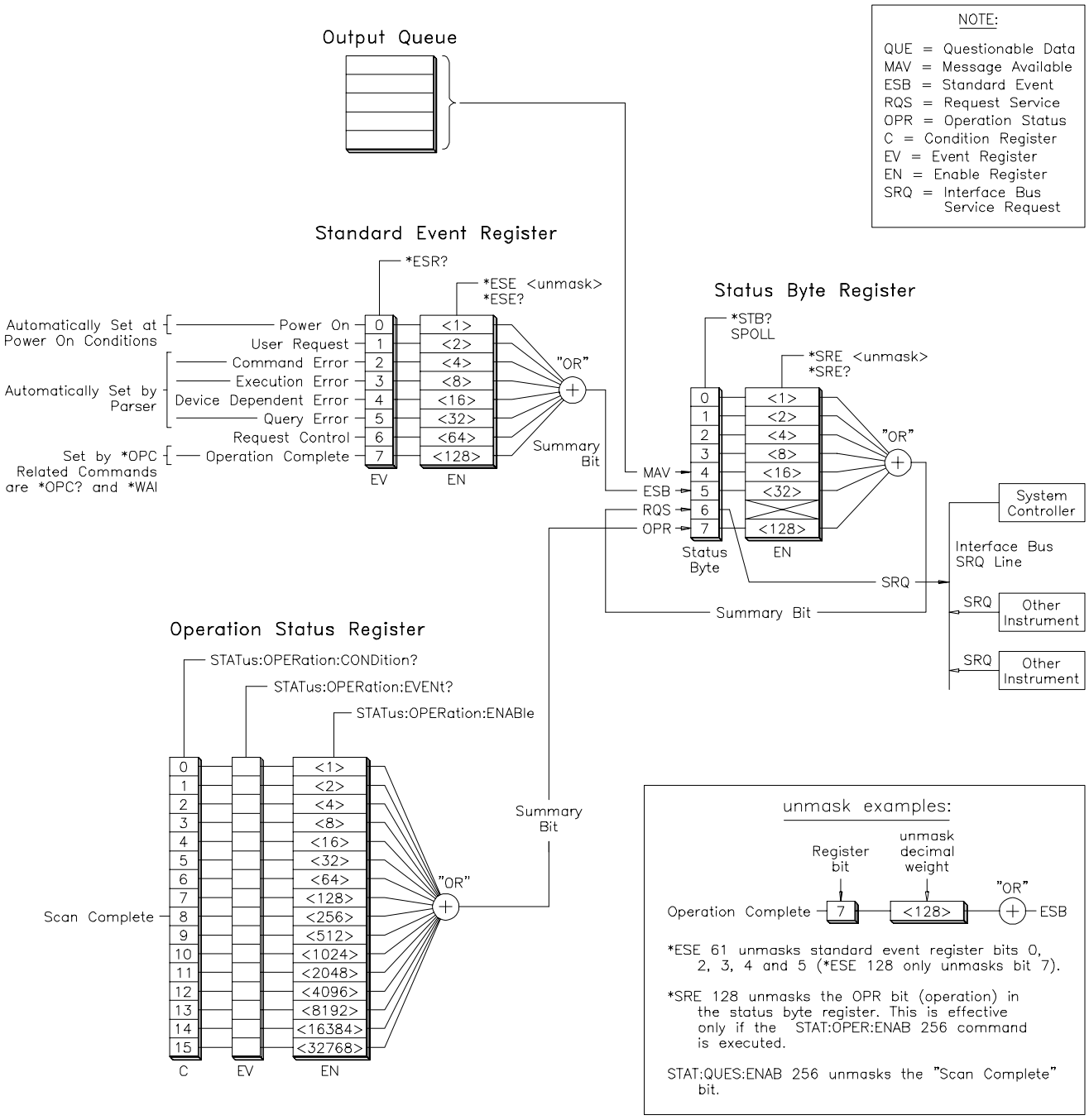


Figure 3-1. HP E1463A Status System Register Diagram

:OPERation :CONDition?

STATus:OPERation:CONDition? returns the state of the Condition Register in the Operation Status Group. The state represents conditions which are part of the instrument's operation. The switch driver does not set bit 8 in this register (see STATus:OPERation[:EVENt]?).

:OPERation:ENABle

STATus:OPERation:ENABle <*unmask*> sets an enable mask to allow events recorded in the Event Register to send a summary bit to the Status Byte Register (bit 7). For switch modules, when bit 8 in the Operation Status Register is set to 1 and that bit is enabled by the STATus:OPERation:ENABle command, bit 7 in the Status Register is set to 1.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
unmask	numeric	0 through 65,535	N/A

Comments

- **Setting Bit 7 of the Status Register:** STATus:OPERation:ENABle 256 sets bit 7 of the Status Register to 1 after bit 8 of the Operation Status Register is set to 1.
- **Related Commands:** [ROUTe:]SCAN

Example

Enabling Operation Status Register Bit 8

```
STAT:OPER:ENAB  256          !Enable bit 8 of the Operation Status
                             Register to be reported to bit 7
                             (OPR) in the Status Register.
```

:OPERation:ENABle?

STATus:OPERation:ENABle? returns the bit value of the Operation Status Register.

Comments

- **Output Format:** Returns a decimal weighted value from 0 to 65,535 indicating which bits are set to true.
- **Maximum Value Returned:** The value returned is the value set by the STAT:OPER:ENAB <*unmask*> command. However, the maximum decimal weighted value used in this module is 256 (bit 8 set to true).

Example

Query the Operation Status Enable Register

```
STAT:OPER:ENAB?              !Query the Operation Status
                             Enable Register.
```

:OPERation[:EVENt]?

STATus:OPERation[:EVENt]? returns which bits in the Event Register (Operation Status Group) are set. The Event Register indicates when there has been a time-related instrument event.

Comments

- **Setting Bit 8 of the Operation Status Register:** Bit 8 (scan complete) is set to 1 after a scanning cycle completes. Bit 8 returns to 0 (zero) after sending the STATus:OPERation[:EVENt]? command.
- **Returned Data after sending the STATus:OPERation[:EVENt]? Command:** The command returns "+256" if bit 8 of the Operation Status Register is set to 1. The command returns "+0" if bit 8 of the Operation Status Register is set to 0.
- **Event Register Cleared:** Reading the Event Register with the STATus:OPERation:EVENt? command clears it.
- **Aborting a scan:** Aborting a scan will leave bit 8 set to 0.
- **Related Commands:** [ROUTe:]SCAN

Example

Reading the Operation Status Register After a Scanning Cycle

```
STAT:OPER?                  !Return the bit values of the
                            Operation Status Register.
read the register value     +256 shows bit 8 is set to 1;
                            +0 shows bit 8 is set to 0.
```

:PRESet

STATus:PRESet affects only the Enable Register by setting all Enable Register bits to 0. It does not affect either the "status byte" or the "standard event status". PRESet does not clear any of the Event Registers.

SYSTem

The SYSTem subsystem returns the error numbers and error messages in the error queue of a switchbox. It can also return the types and descriptions of modules (cards) in a switchbox.

Subsystem Syntax

```
SYSTem
    :CDEScription? <number>
    :CPON <number> | ALL
    :CTYPe? <number>
    :ERRor?
```

:CDEScription?

SYSTem:CDEScription? <*number*> returns the description of a selected module (card) in a switchbox.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
number	numeric	1 through 99	N/A

Comments

- **Form C Switch Module Description:** The SYSTem:CDEScription? command returns:

  "32 Channel General Purpose Relay"

Example

Reading the Description of a Card #1 Module

```
SYST:CDES? 1                    !Return the description.
```

:CPON

SYSTem:CPON <*number*> | ALL sets the selected module (card) in a switchbox to its power-on state.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
number	numeric	1 through 99	N/A

Comments

- **Form C Switch Module Power-on State:** The power-on state is all channels (relays) open. Note that SYSTem:CPON ALL and *RST opens all channels of all modules in a switchbox, while SYSTem:CPON <*number*> opens the channels in only the module (card) specified in the command.

Example

Setting Card #1 Module to its Power-on State

```
SYST:CPON 1                     !Set card #1 to power-on state.
```

:CTYPe?

SYSTem:CTYPe? <*number*> returns the module (card) type of a selected module in a switchbox.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
number	numeric	1 through 99	N/A

Comments

- **HP E1463A Form C Switch Module Model Number:** The SYSTem:CTYPe? <*number*> command returns:

  ```
  HEWLETT-PACKARD,E1463A,0,A.04.00
  ```

  where the 0 after E1463A is the module serial number (always 0) and A.04.00 is an example of the module revision code number.

Example

Reading the Model Number of a Card #1 Module

```
SYST:CTYP? 1                    !Return the model number.
```

:ERRor?

SYSTem:ERRor? returns the error numbers and corresponding error messages in the error queue of a switchbox. See Appendix C for a listing of switchbox error numbers and messages.

Comments

- **Error Numbers/Messages in the Error Queue:** Each error generated by a switchbox stores an error number and corresponding error message in the error queue. The error message can be up to 255 characters long.

- **Clearing the Error Queue:** An error number/message is removed from the queue each time the SYSTem:ERRor? command is sent. The errors are cleared first-in, first-out. When the queue is empty, each following SYSTem:ERRor? command returns `+0, "No error"`. To clear all error numbers/messages in the queue, execute the *CLS command.

- **Maximum Error Numbers/Messages in the Error Queue:** The queue holds a maximum of 30 error numbers/messages for each switchbox. If the queue overflows, the last error number/message in the queue is replaced by `-350, "Too many errors"`. The least recent error numbers/messages remain in the queue and the most recent are discarded.

Example

Reading the Error Queue

```
SYST:ERR?                       !Query the error queue.
```

TRIGger

The TRIGger command subsystem controls the triggering operation of Form C switch modules in a switchbox.

Subsystem Syntax

```
TRIGger
    [:IMMediate]
    :SOURce <source>
    :SOURce?
```

[:IMMediate]

TRIGger[:IMMediate] causes a trigger event to occur when the defined trigger source is TRIGger:SOURce BUS or TRIGger:SOURce HOLD.

Comments

- **Executing the TRIGger[:IMMediate] Command:** A channel list must be defined with [ROUTe:]SCAN <*channel_list*> and an INITiate[:IMMediate] command must be executed before TRIGger[:IMMediate] will execute.
- **BUS or HOLD Source Remains:** If selected, the TRIGger:SOURce BUS or TRIGger:SOURce HOLD commands remain in effect after triggering a switchbox with the TRIGger[:IMMediate] command.
- **Related Commands:** INITiate, [ROUTe:]SCAN

Example

Advancing Scan Using TRIGger Command

This example uses the TRIGger command to advance the scan of a single-module switchbox from channel 00 through 03. Since TRIGger:SOURce HOLD is set, the scan is advanced one channel each time TRIGger is executed.

```
TRIG:SOUR  HOLD          !Set trigger source to HOLD.
SCAN  (@100:103)         !Define channel list.
INIT                     !Begin scan, close channel 00.
loop statement           !Start count loop.
TRIG                     !Advance scan to next channel.
increment loop           !Increment loop count.
```

:SOURce

TRIGger:SOURce <*source*> specifies the trigger *source* to advance the channel list during scanning.

Parameters

Parameter Name	Parameter Type	Parameter Description	Default Value
BUS	discrete	*TRG or GET command	IMM
EXTernal	discrete	"Trig In" port	IMM
HOLD	discrete	Hold Triggering	IMM
IMMediate	discrete	Immediate Triggering	IMM
TTLTrg*n*	numeric	TTL Trigger bus line 0 - 7	IMM

Comments

- **Enabling the Trigger Source:** The TRIGger:SOURce command only selects the trigger *source*. The INITiate[:IMMediate] command enables the trigger source.
- **Using the TRIGger Command:** You can use TRIGger[:IMMediate] to advance the scan when TRIGger:SOURce BUS or TRIGger:SOURce HOLD is selected.
- **Using External Trigger Inputs:** With TRIGger:SOURce EXTernal selected, only one switchbox at a time can use the external trigger input at the HP E1406A "Trig In" port. The trigger input is assigned to the first switchbox requesting the external trigger source (with a TRIGger:SOURce EXTernal command).
- **Assigning External Trigger:** A switchbox assigned with TRIGger:SOURce EXTernal remains assigned to that source until the switchbox trigger source is changed to BUS, HOLD, or IMMediate. When the source is changed, the external trigger source is available to the next switchbox requesting it (with a TRIGger:SOURce EXTernal command). If a switchbox requests an external trigger input already assigned to another switchbox an error is generated.
- **Using Bus Triggers:** To trigger the switchbox with TRIGger:SOURce BUS selected, use the IEEE 488.2 common command *TRG or the HP-IB Group Execute Trigger (GET) command.
- **"Trig Out" Port Shared by Switchboxes:** See the OUTPut command on page 59.
- **Related Commands:** ABORt, [ROUTe:]SCAN, OUTPut
- ***RST Condition:** TRIGger:SOURce IMMediate

Example Scanning Using External Triggers

This example uses external triggering (TRIG:SOUR EXT) to scan channels 00 through 03 of a single-module switchbox. The trigger source to advance the scan is the input to the "Trig In" on the HP E1406A command module. When INIT is executed, the scan is started and channel 00 is closed. Then, each trigger received at the "Trig In" port advances the scan to the next channel.

```
TRIG:SOUR  EXT                  !Select external triggering.
SCAN  (@100:103)                !Scan channels 00 through 03.
INIT                            !Begin scan, close channel 00.
trigger externally              !Advance scan to next channel.
```

Example Scanning Using Bus Triggers

This example uses bus triggering (TRIG:SOUR BUS) to scan channels 00 through 03 of a single-module switchbox. The trigger source to advance the scan is the *TRG command (as set with TRIGger:SOURce BUS). When INIT is executed, the scan is started and channel 00 is closed. Then, each *TRG command advances the scan to the next channel.

```
TRIG:SOUR  BUS                  !Select interface (bus) triggering.
SCAN  (@100:103)                !Scan channels 00 through 03.
INIT                            !Begin scan, close channel 00.
loop statement                  !Loop to scan all channels.
*TRG                            !Advance scan using bus triggering.
increment loop                  !Increment loop count.
```

:SOURce?

TRIGger:SOURce? returns the current trigger source for the switchbox. Command returns BUS, EXT, HOLD, IMM, or TTLT for sources BUS, EXTernal, HOLD, IMMediate, or TTLTrg*n*, respectively.

Example Querying the Trigger Source

This example sets external triggering and queries the trigger source. Since external triggering is set, TRIG:SOUR? returns "EXT".

```
TRIG:SOUR  EXT                  !Set external trigger source.
TRIG:SOUR?                      !Query trigger source.
```

IEEE 488.2 Common Command Reference

The following table lists the IEEE 488.2 Common (*) Commands that apply to the HP E1463A module. The operation of some of these commands is described in earlier in this manual. For more information on Common Commands, refer to the *HP E1406A Command Module User's Manual* or the *ANSI/IEEE Standard 488.2-1987*.

Command	Title	Description
*IDN?	Identification Query	Returns Identification String of the Switchbox.
*RST	Reset	Opens all channels, and sets the module to a known state.
*TST?	Self-Test Query	Returns +0 if self-test passes. Returns +cc01 for firmware error. Returns +cc02 for bus error. Returns +cc10 if an interrupt was expected but not received. Returns +cc11 if the busy bit was not held for 10 msec.
*OPC	Operation Complete	Sets the Request for OPC flag when all pending operations have completed. Also sets OPC bit in the Standard Event Register.
*OPC?	Operation Complete Query	Returns a "1" to the output queue when all pending operations have completed. Used to synchronize between multiple instruments.
*WAI	Wait to Continue	Prevents an instrument from executing another command until the operation caused by the previous command is finished. Since all instruments normally perform sequential operations, executing this command causes no change.
*CLS	Clear Status Register	Clears all Status Registers (see STATus:OPERation[:EVENt]?).
*ESE	Event Status Enable	Enables Status Register bits.
*ESE?	Event Status Enable Query	Queries the current contents in the Event Status Register.
*ESR?	Event Status Register Query	Queries and clears current the contents in the Event Status Register.
*SRE	Service Request Enable	Used to set the Service Request Enable Register bits, and corresponding Serial Poll Status Register bits, to generate a service request.
*SRE?	Service Request Enable Query	Queries the current contents in the Service Request Enable Register.
*STB?	Read Status Byte Query	Queries the current contents in the Status Byte Register.
*TRG	Trigger	Triggers the switchbox to advance the scan when scan is enabled and trigger source is TRIGger:SOURce BUS.
*RCL	Recall Instrument State	Recalls previously stored configuration.
*SAV	Save Instrument State	Stores the current configuration in specified memory.

SCPI Command Quick Reference

The following table summarizes the SCPI commands for the Form C switch.

Command Subsystem	Command/Parameter	Description
ABORt	ABORt	Abort a scan in progress.
ARM	:COUNt <*number*> *MIN* | *MAX* :COUNT? [MIN | MAX]	Multiple scans per INIT command. Query number of scans.
DISPlay	:MONitor:CARD <*number*> | AUTO :MONitor[:STATe] <*mode*>	Selects the module in a switchbox to be monitored. Turns monitor mode on or off.
INITiate	:CONTinuous <*mode*> :CONTinuous? [:IMMediate]	Enables/Disables continuous scanning. Query continuous scan state. Starts a scanning cycle.
OUTPut	:EXTernal[:STATe] <*mode*> :EXTernal[:STATe]? [:STATe] <*mode*> [:STATe]? :TTLTrg*n*[:STATe] <*mode*> :TTLTrg*n*[:STATe]?	Enables/Disables "Trig Out" pulse. Query port enable state. Enables/Disables "Trig Out" pulse. Query port enable state. Enables/Disables TTL Trigger bus line pulse. Query TTL Trigger Bus line state.
[ROUTe:]	CLOSe <*channel_list*> CLOSe? <*channel_list*> OPEN <*channel_list*> OPEN? <*channel_list*> SCAN <*channel_list*>	Close channel(s). Query channel(s) closed. Open channel(s). Query channel(s) opened. Define channels for scanning.
STATus	:OPERation:CONDition? :OPERation:ENABle <*unmask*> :OPERation:ENABle? :OPERation[:EVENt]? :PRESet	Returns status of the Condition Register. Enables the Operation Status Register to set a bit in the Status Register. Query the contents in the Operation Status Register. Returns status of Operation Status Register. Sets Enable Register to 0.
SYSTem	:CDEScription? <*number*> :CTYPe? <*number*> :CPON <*number*> | ALL :ERRor?	Returns description of module in switchbox. Returns the module type. Sets specified module to its power-on state. Returns error number/message to error queue.
TRIGger	[:IMMediate] :SOURce BUS :SOURce EXTernal :SOURce HOLD :SOURce IMMediate :SOURce TTLTrg*n* :SOURce?	Causes a trigger to occur. Trigger source is *TRG. Trigger source is "Trig In". Hold off triggering. Continuous (internal) triggering. Trigger source is TTL trigger bus line (0 - 7). Query scan trigger source.

Appendix A
Specifications

General

Module Size/Device Type:
C-Size VXIbus, Register based, A16/D16,
Interrupter (levels 1-7, jumper selectable)

Relay Life (typical):

Condition	Number of Operations
NO Load	5×10^7
250 Vac, 2 A, Resistive	10^6
250 Vac, 5 A, Resistive	10^5
250 Vac, 2 A, p.f. = 0.4	2.5×10^5
250 Vac, 5 A, p.f. = 0.4	3.5×10^4
30 Vdc, 1 A, Resistive	$>10^6$
30 Vdc, 5 A, Resistive	10^5
30 Vdc, 1 A, L/R = 7 msec	$>10^6$
30 Vdc, 5 A, L/R = 7 msec	10^5

NOTE: Relays are subject to normal wear out based on the number of operations.

Terminals:
Screw type, maximum wire size 16 AWG

Power Requirements:

Voltage:	+5	+12
Peak Module Current (A):	0.10	0.60*
Dynamic Module Current (A):	0.10	0.01

Watts/slot: 10 W

Cooling/slot: .08 mm H_2O @ 0.42 Liter/sec for 10°C rise

Operating Temperature: 0 - 55°C

Operating Humidity: 65% RH, 0 - 40°C

Input Characteristics

Maximum Input Voltage:
250 Vdc or ac_{rms} Terminal to Terminal
250 Vdc or ac_{rms} Terminal to Chassis

Maximum Current per Channel (non-inductive):
5 Adc or ac_{rms}

Maximum Switchable Power per Channel:
150 W dc; 1,250 VA per switch
1,500 W dc; 12,500 VA per module

DC Performance

Thermal Offset per Channel:
<7µV (<3 µV typical)

Closed Channel Resistance:
>100 mA: <0.250Ω
(< 2Ω at end of relay life)
<100 mA: <20Ω

Insulation Resistance (between any two points):
$>10^8$ Ω (at ≤40°C, ≤95% RH)
$>10^9$ Ω (at ≤25°C,≤ 40% RH)

AC Performance

Capacitance:
< 30 pF (Channel to Channel)
<40 pF (Channel to Common)
<25 pF (Common to Guard)

Bandwidth (-3dB): >10 MHz (typical)

Crosstalk(dB) (for Z1 = Zs = 50Ω):

Frequency	<10kHz	<100kHz	<1MHz
Channel to Channel	<-83	<-63	<-43
Common to NO or NC	<-80	<-60	<-40
Module to Module	<-100	<-100	<-90

* Absolute worst case when all relays are closed simultaneously

Relay Life

Relay Life

Electromechanical relays are subject to normal wear-out. Relay life depends on several factors. The effects of loading and switching frequency are briefly discussed below:

Relay Load. In general, higher power switching reduces relay life. In addition, capacitive/inductive loads and high inrush currents (for example, turning on a lamp or starting a motor) reduces relay life. *Exceeding specified maximum inputs can cause catastrophic failure.*

Switching Frequency. Relay contacts heat up when switched. As the switching frequency increases, the contacts have less time to dissipate heat. The resulting increase in contact temperature also reduces relay life.

End-of-Life Detection

A preventive maintenance routine can prevent problems caused by unexpected relay failure. The end of the life of the relay can be determined by using one or more of the three methods described below. The best method (or combination of methods), as well as the failure criteria, depends on the application in which the relay is used.

Contact Resistance. As the relay begins to wear out, its contact resistance increases. When the resistance exceeds a predetermined value, the relay should be replaced.

Stability of Contact Resistance. The stability of the contact resistance decreases with age. Using this method, the contact resistance is measured several (5-10) times, and the variance of the measurements is determined. An increase in the variance indicates deteriorating performance.

Number of Operations. Relays can be replaced after a predetermined number of contact closures. However, this method requires knowledge of the applied load and life specifications for the applied load.

Replacement Strategy

The replacement strategy depends on the application. If some relays are used more often, or at a higher load, than the others, the relays can be individually replaced as needed. If all the relays see similar loads and switching frequencies, the entire circuit board can be replaced when the end of relay life approaches. The sensitivity of the application should be weighed against the cost of replacing relays with some useful life remaining.

Note Relays that wear out normally or fail due to misuse should not be considered defective and are not covered by the product's warranty.

Appendix B
Register-Based Programming

About This Appendix

The HP E1463A 32-Channel, 5 Amp, Form C Switch Module is a register-based device. When a SCPI command is sent to the Form C module, the HP E1406A command module parses the command and programs the Form C switch at the register level.

Register-based programming is a series of **reads** and **writes** directly to the Form C switch module registers. Writing directly to the registers can increase throughput speed since it eliminates the command parsing and allows the use of an embedded controller. This appendix includes information on the following:

Register Addressing

Register addresses for register-based devices are located in the upper 25% of VXI A16 address space. Every VXI module (up to 256 devices) is allocated a 32 word (64-byte) block of addresses.

Figure B-1 shows the register address location within A16. Figure B-2 shows the location of A16 address space in the HP E1406A command module.

When you are reading or writing to a Form C switch module register, a hexadecimal or decimal register address needs to be specified. This address consists of a base address plus a register offset:

Register Address = Base Address + Register Offset

The Base Address

The base address used in register-based programming depends on whether the A16 address space in the HP E1406A command module is outside or inside the HP command module.

A16 Address Space Outside the Command Module

When the command module is not part of your VXIbus system (Figure B-1), the Form C switch module's base address depends on the command module used.[1]

$C000_{16} + (LADDR * 64)_{16}$

or

$49{,}152 + (LADDR * 64)$

Where $C000_{16}$ (49,152) is the starting location of the VXI A16 addresses, LADDR is the Form C switch module's logical address, and 64 is the number of address bytes per register-based module. For example, the Form C module's factory set logical address is 120. If the address is not changed, the Form C module will have the following base address:

$C000_{16} + (120 * 64)_{16} =$ **$DE00_{16}$**

or (decimal)

$49{,}152 + (120 * 64) =$ **56,832**

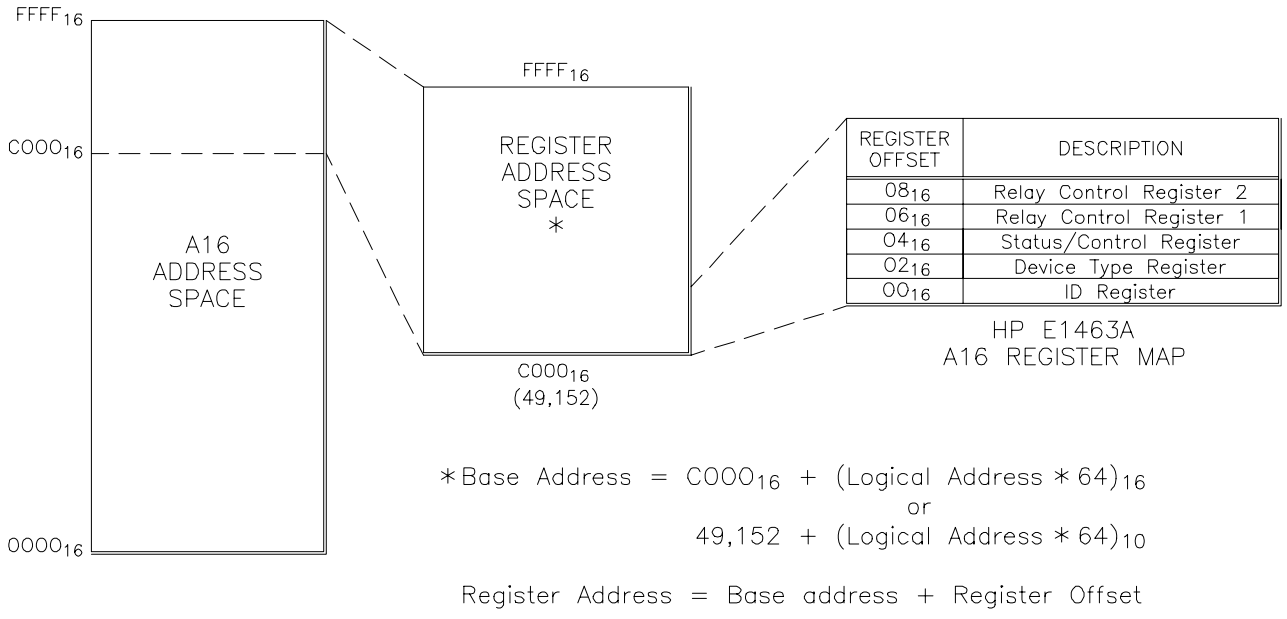


Figure B-1. Registers within A16 Address Space

1 The "16" at the end of the address indicates a hexadecimal number.

A16 Address Space Inside the Command Module

When the A16 address space is inside the command module (Figure B-2), the Form C module's base address is computed as follows:

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

or

2,080,768 + (LADDR * 64)

Where $1FC000_{16}$ (2,080,768) is the starting location of the register addresses, LADDR is the Form C module's logical address, and 64 is the number of address bytes per VXI module. Again, the Form C's factory set logical address is 120. If the address is not changed, the Form C modules will have the following base address:

$1FC000_{16} + (120 * 64)_{16} =$ **$1FDE00_{16}$**

or

2,080,768 + (120 * 64) = **2,088,448**

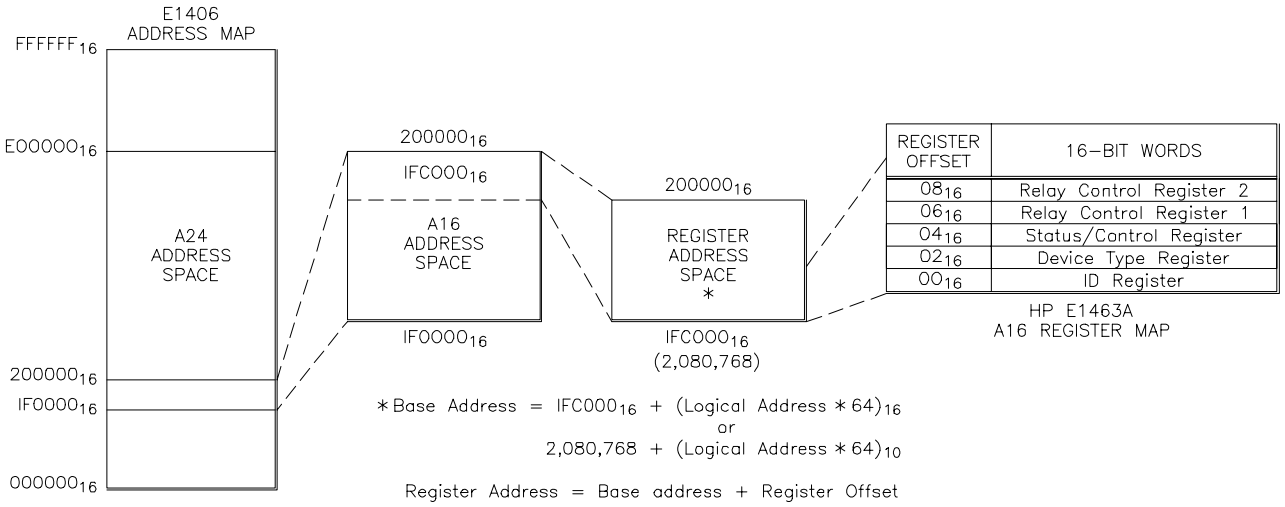


Figure B-2. Registers within HP Command Module A16 Address Space

Register Offset

The register offset is the register's location in the block of 64 address bytes. For example, the Form C module's Status Register has an offset of 04_{16} (see the next section). When you write a command to this register, the register offset is added to the base address to form the following register address:

$1FDE00_{16} + 04_{16} =$ **$1FDE04_{16}$**

or

2,088,448 + 4 = **2,088,452**

Register Descriptions

The Form C switch modules contain 2 READ registers, 1 READ/WRITE register, and 2 WRITE registers. This section describes each Form C module register.

Reading and Writing to the Registers

Example programs are provided at the end of this appendix that show how to read and write to these registers. You can read or write to the following Form C switch module registers:

- Manufacturer ID Register (base + 00_{16}) (read)
- Device Type Register (base + 02_{16}) (read)
- Status/Control Register (base + 04_{16}) (read or write)
- Relay Control Register for Channels 00 - 15 (base + 06_{16}) (write)
- Relay Control Register for Channels 16 - 31 (base + 08_{16}) (write)

Each of these registers is discussed in the following sections.

The Manufacturer Identification Register

The Manufacturer Identification Register is at offset address 00_{16} and returns $FFFF_{16}$. This shows Hewlett-Packard as the manufacturer and the module is an A16 register-based module. This register cannot be written to.

base + 00_{16}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	Manufacturer ID - returns $FFFF_{16}$ in Hewlett-Packard A16 register-based card															

The Device Type Register

The Device Type Register is at offset address 02_{16} and returns 0121_{16} if you have an HP E1463A Form C Switch Module. The Device Type Register cannot be written to.

base + 02_{16}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	0121_{16}															

The Status/Control Register

The Status/Control Register is at offset address 04_{16} and informs the user about the module's status and configuration.

base + 04_{16}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Not Used									E	Not Used					R
Read	X	MS	Not Used						B	E	X	X	1	1	X	X

Reading the Status/ Control Register

When reading the status/control register, the following bits are of importance:

Enable (bit 6) 0 indicates that the interrupt is enabled. The interrupt generated after a channel has been closed can be disabled. Bit 6 of this register is used to inform the user of the interrupt status.

Busy (bit 7) 0 indicates that the module is busy. Each relay requires about 10 ms execution time during which time the Form C switch is busy. Bit 7 of this register is used to inform the user of a busy condition.

Modid Select (bit 14) 0 indicates that the module has been selected by MODID (module ID), and a 1 indicates it has not.

For example, if the Form C switch module is not busy (bit 7 = 1) and the interrupt is enabled (bit 6 = 0), then a read of the Status/Control Register (base + 04_{16}) returns FFBF.

Writing to the Status/ Control Register

You can write to bits 0 and 6. See the following explanations:

Soft Reset (bit 0) Writing a "1" to this bit soft resets the module.

Enable (bit 6) Writing a "1" to this bit disables the interrupt function of the module.

Note

When writing to the registers it is necessary to write "0" to bit 0 after the reset has been performed before any other commands can be programmed and executed. SCPI commands take care of this automatically.

Typically, interrupts are only disabled to "peek-poke" a module. Refer to the operating manual of the command module used before disabling the interrupt. Writing a "1" to bit 0 resets the switch (all channels open).

The Relay Control Register

There are two relay control registers. These registers are used to connect the common (C) to the normally open (NO) terminal. When reading these registers, $FFFF_{16}$ is always returned.

- Relay Control Register 1 (base + 06_{16})
- Relay Control Register 2 (base + 08_{16})

The numbers shown in the register maps indicate the channel number to be written to. Writes to the Relay Control Registers enable you to open or close the desired channel. For example, write a "1" to bit 2 of the Relay Control Register 06_{16} to close channel 02:

WRITEIO -16, (2088448 + 06);4

where, the 4 represents 100 in hexadecimal.

Relay Control Register Channels 00 - 15

base + 06_{16}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	CH15	CH14	CH13	CH12	CH11	CH10	CH09	CH08	CH07	CH06	CH05	CH04	CH03	CH02	CH01	CH00
Read	Always Returns $FFFF_{16}$															

Relay Control Register Channels 16 - 31

base + 08_{16}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
Read	Always Returns $FFFF_{16}$															

Programming Examples

The following sections provide examples in both HP BASIC and HP-UX, C. These examples support the following configuration:

- **Mainframe:** HP 75000 Series C (HP E1401A)
- **Controller:** HP V/360 (HP E1480A) w/Resource Manager and Slot 0
- **Programming Language:** HP BASIC/HP-UX, C
- **Switch Card:** HP 32-Channel, 5 Amp Form C Switch Module (HP E1463A)

Reading the Registers

The following examples read the module's Manufacturer ID, Device Type, and Status Registers on the Form C switch.

HP BASIC

```
10   !************************************************
20   !****                    READREG                 *****
30   !************************************************

40   OPTION BASE 1
50   !Set up arrays to store register names and addresses.
60   DIM Reg_name$(1:3)[32], Reg_addr(1:3)
70   !
80   !Read register names and addresses into the arrays.
90   READ Reg_name$(*)
100  READ Reg_addr(*)
110  !
120  !Set base address variable.
130  Base_addr = DVAL("DE00",16)
140  !
150  !Map the A16 address space in the V/360.
160  !
170  CONTROL 16,25;2
180  !Call the subprogram Read_regs.
190  Read_regs(Base_addr, Reg_name$(*), Reg_addr(*))
200  !
210  DATA Identification register, Device register, Status register
220  DATA 00, 02, 04
230  END
  .
  .
  .
300  !This subprogram steps through a loop that reads each register and prints
310  !its contents.
320  SUB Read_regs(Base_addr, Reg_name$(*), Reg_addr(*))
330  !
340  For Number = 1 to 3
350     Register = READIO(-16,Base_addr + Reg_addr(number))
360     PRINT Reg_name$(number); " = "; IVAL$(Register,16)
370  Next Number
380  SUBEND
```

HP-UX, C

```
/*********************************************************/
/***                                                   ***/
/*********************************************************/
#include     <sys/vxi.h>          /*source file for HP V/360 VXI drivers*/
#include     <fcntl.h>
#include     <stdio.h>

#define logical_address 120      /*logical address of the form c module*/

int fd;
typedef unsigned short word;
typedef struct dev_regs{         /*set up pointers*/
                              unsigned short id_reg;
                              unsigned short device_type;
                              unsigned short status_reg;
                              unsigned short bank0_channels;
} DEV_REGS;

main( )
{
                                 /*open the HP V/360 VXI interface*/
fd=open("/dev/vxi/primary",O_RDWR);
if (fd){
          perror("open");
          exit(1);
 }
                                 /*retrieve the A16 pointers*/
dev=(struct dev_regs *)vxi_get_a16_addr(fd,logical_address);

                                 /*sub to read the registers*/
read_reg(dev);

}                                /*END of main program*/

/*SUB READ_REG*/

int   read_reg(reg_ptr)
DEV_REGS *reg_ptr;
{                                /*read the id register*/

printf("\n ID Register = 0x%x\n",reg_ptr->id_reg);

                                 /*read the device type register*/
printf("\n DEVICE TYPE Register = 0x%x\n",reg_ptr->device_type);

                                 /*read the status register*/
printf("\n STATUS Register = 0x%x\n",reg_ptr->status_reg);
return;
}
```

Making Measurements

The following examples close bit 1 on bank 0, wait for a measurement to be made, and then open the channel.

You must insert your own programming code for the measurement part of this program. If you are using the HP E1411B, see the *HP E1326B/E1411B 5½-Digit Multimeter User's Manual* for programming examples.

HP BASIC

```
 10  !************************************************
 20  !*****                MAKEMEAS                *****
 30  !************************************************

 40  OPTION BASE 1
 50  !Set up arrays to store register names and addresses.
 60  DIM Reg_name$(1:1)[32], Reg_addr(1:1)
 70  !
 80  !Read register names and address into the arrays.
 90  READ Reg_name$(*)
100  READ Reg_addr(*)
110  !
120  !Set base address variable.
130  Base_addr = DVAL("DE00",16)
140  !
150  !Map the A16 address space in the V/360.
160  CONTROL 16,25;2
170  !Call the subprogram Make_meas.
180  Make_meas(Base_addr, Reg_addr(*))
190  !
200  DATA Bank0 channels register
210  DATA 06
220  END
  .
  .
  .
280  !This subprogram closes bit 1 of bank0 channels, waits for the channel
290  !to be closed, makes a measurement, and then opens the relay.
300  SUB Make_meas(Base_addr, Reg_addr(*))
310  !
320  WRITEIO -16, Base_addr + Reg_addr(1); 1
330  REPEAT
340  UNTIL BIT(READIO(-16,Base_addr+4),7)
  .
  .  !Make Measurements.
  .
380  WRITEIO -16, Base_addr + Reg_addr(1);0
390  SUBEND
```

HP-UX, C

```
/*********************************************************/
/***                                   readreg.c                                  ***/
/*********************************************************/
#include     <time.h>
#include     <sys/vxi.h>              /*source file for HP V/360 VXI drivers*/
#include     <fcntl.h>
#include     <stdio.h>

#define logical_address 120           /*logical address of the form c module*/

int fd;
typedef unsigned short word;
typedef struct dev_regs{              /*set up pointers*/
                              unsigned short id_reg;
                              unsigned short device_type;
                              unsigned short status_reg;
                              unsigned short bank0_channels;
} DEV_REGS;

main( )
{
                                      /*open the HP V/360 VXI interface*/
fd=open("/dev/vxi/primary",O_RDWR);
if (fd){
               perror("open");
               exit(1);
 }
                                      /*retrieve the A16 pointers*/
dev=(struct dev_regs *)vxi_get_a16_addr(fd,logical_address);

                                      /*sub to verify the time to close the switch*/
ver_time( );

                                      /*sub to close switch and make measurement*/
make_meas(dev);
}                                     /*END of main program*/

/*SUB VER_TIME*/

ver_time( )
{
struct timeval first,
                second,
                lapsed;

struct timezone tzp;
```

Continued on Next Page

```
gettimeofday(&first,&tzp);
for (j=0; j<=10000; j ++);
gettimeofday ($second,&tzp);

if (first.tv_usec > second.tv_usec)
      {
      second.tv_usec +=1000000;
      second.tv_sec--;
}


lapsed.tv_usec = second.tv_usec - first.tv_usec;
lapsed.tv_sec = second.tv_sec - first.tv_sec;

printf("Elapsed time for closing a channel is:  %ld sec %ld usec \n",
lapsed.tv_sec, lapsed.tv_usec);
}

/*SUB MAKE_MEAS*/

int make_meas(reg_ptr)
DEV_REGS *reg_ptr;
{
                                        /*close bit 1 of bank0 */
reg_ptr->bank0_channels=0x0001;
for (j=0; j<=10000; j ++);              /*wait for switch to close*/
printf("\n Making Measurement");
            .
            .                           /*make measurements*/
            .
                                        /*open bit 1 of bank0*/
reg_ptr->bank0_channels=0x0000;
return;
}
```

Note The sub ***ver_time*** allows time for switch closures. This sub should print a time around 10 ms. If the time is less, you must change the value of **j** in the for loop. For example, instead of 10000, you might have to use 12000.

Scanning Channels

The following examples scan through the bank 0 channels (closing one switch at a time) and make measurements between switch closures.

Again, you must insert your own programming code for the measurement part of this program. If you are using the HP E1411B, see the *HP E1326B/E1411B 5½-Digit Multimeter User's Manual* for programming examples.

HP BASIC

```
10  !************************************************
20  !*****                SCANNING                *****
30  !************************************************
40  OPTION BASE 1
50  !Set up arrays to store register names and addresses.
60  DIM Reg_name$(1:1)[32], Reg_addr(1:1)
70  !
80  !Read register names and addresses into the arrays.
90  READ Reg_name$(*)
100 READ Reg_addr(*)
110 !Set base address variable.
120 Base_addr = DVAL("DE00",16)
130 !
140 !Map the A16 address space in the V/360.
150 CONTROL 16,25;2
160 !Call the subprogram Scan_meas.
170 Scan_meas(Base_addr, Reg_addr(*))
180 !
190 DATA Bank0 channels register
200 DATA 06
210 END
  .
  .
  .
270 !This subprogram sets all bits in bank0 open then scan through bank 0,
280 !closing one channel at a time (waits for the channel to be closed) so a
290 !measurement can be made.
300 SUB Scan_meas(Base_addr, Reg_addr(*))
310 !
320 WRITEIO -16, Base_addr + Reg_addr(1);0
330 FOR I = 0 to 15
340    WRITEIO -16, Base_addr + Reg_addr(1);2^I
350    REPEAT
360    UNTIL BIT(READIO(-16,Base_addr+4),7)
370    PRINT "Making Measurement"
  .
  .    !Make Measurements.
  .
420 NEXT I
430 WRITEIO -16,Base_addr + Reg_addr(1);0
440 SUBEND
```

HP-UX, C

```
/*********************************************************/
/***                                scanning.c                                ***/
/*********************************************************/
#include     <time.h>
#include     <math.h>                /*file to perform math functions*/
#include     <sys/vxi.h>             /*source file for HP V/360 VXI drivers*/
#include     <fcntl.h>
#include     <stdio.h>

#define logical_address 120          /*logical address of the form c module*/
#define lastch15

int fd, i, reg;
double y;
typedef unsigned short word;
typedef struct dev_regs{             /*set up pointers*/
                              unsigned short id_reg;
                              unsigned short device_type;
                              unsigned short status_reg;
                              unsigned short bank0_channels;
} DEV_REGS;
main( )
{
                                     /*open the HP V/360 VXI interface*/
fd=open("/dev/vxi/primary",O_RDWR);
if (fd){
          perror("open");
          exit(1);
 }                                   /*retrieve the A16 pointers*/
dev=(struct dev_regs *)vxi_get_a16_addr(fd,logical_address);

                                  /*sub to verify the time to close the switch*/
ver_time( );
                        /*sub to close a set of switches and make measurements*/
scan_meas(dev);
}                                    /*END of main program*/
/*SUB VER_TIME*/

ver_time( )
{
struct timeval first,
               second,
               lapsed;

struct timezone tzp;
```

Continued on Next Page

```
gettimeofday(&first,&tzp);
for (j=0; j<=10000; j ++);
gettimeofday ($second,&tzp);
if (first.tv_usec > second.tv_usec)
      {
      second.tv_usec +=1000000;
      second.tv_sec--;
}

lapsed.tv_usec = second.tv_usec - first.tv_usec;
lapsed.tv_sec = second.tv_sec - first.tv_sec;

printf("Elasped time for closing a channel is:  %ld sec %ld usec \n",
lapsed.tv_sec, lapsed.tv_usec);
}

/*SUB SCAN_MEAS*/
int scan_meas(reg_ptr)
DEV_REGS *reg_ptr;
{
                                          /*set bank0 to 000 */
reg_ptr->bank0_channels=0x000;
i=0;
for (i=0;i=lastch;i ++)
      {
      y=i;
      reg=pow(2.0,y);
      reg_ptr-bank0_channels=reg;
      for (j=0; j<=10000; j ++);       /*wait for switch to be closed*/
      printf("\n Making Measurement");
            .
            .                         /*make measurements*/
            .
      }
return;
}
```

Note The sub **ver_time** allows time for the switches to close. The program should print a time around 10 ms. If the time is less, you must change the value of **j** in the for loop. For example, instead of 10000, you might have to use 12000.

Note The **math.h** include file requires an **-lm** option when compiling the above program.

Appendix C
Form C Switch Error Messages

Table C-1 lists the error messages associated with the Form C switch module programmed by SCPI. See the appropriate mainframe manual for a complete list of error messages.

Table C-1. Form C Switch Error Messages

Number	Title	Potential Cause(s)
-109	Missing Parameter	Sending a command requiring a channel list without the channel list.
-211	Trigger Ignored	Trigger received when scan not enabled. Trigger received after scan complete. Trigger too fast.
-213	INIT Ignored	Attempting to execute an INIT command when a scan is already in progress.
-224	Illegal Parameter Value	Attempting to execute a command with a parameter not applicable to the command.
-310	System Error	Too many characters in the channel list expression.
+ 1500	External Trigger Source Already Allocated	Assigning an external trigger source to a switchbox when the trigger source has already been assigned to another switchbox.
+ 2000	Invalid Card Number	Addressing a module (card) in a switchbox that is not part of the switchbox.
+ 2001	Invalid Channel Number	Attempting to address a channel of a module in a switchbox that is not supported by the module (for example, channel 99 of a multiplexer module).
+ 2006	Command Not Supported On This Card	Sending a command to a module (card) in a switchbox that is unsupported by the module.
+ 2008	Scan List Not Initialized	Executing an INIT command without a channel list defined.
+ 2009	Too Many Channels In Channel List	Attempting to address more channels than available in the switchbox.
+ 2011	Empty Channel List	Channel list contains no valid channels.
+ 2012	Invalid Channel Range	Invalid channel(s) specified in SCAN <*channel_list*> command. Attempting to begin scanning when no valid *channel_list* is defined.
+ 2600	Function Not Supported On This Card	Sending a command to a module (card) in a switchbox that is not supported by the module or switchbox.

Index

A

B

C

D

E

P

Q

R

S

T

U

V

W