Agilent Technologies E1442A 64-Channel Form C Switch Module User's Manual

Agilent Technologies

Manual Part Number: E1442-90003
Printed in U.S.A. E1000

Notes:

AGILENT TECHNOLOGIES WARRANTY STATEMENT

AGILENT PRODUCT: E1442A 64-Channel Form C Switch Module **DURATION OF WARRANTY:** 3 years

1. Agilent Technologies warrants Agilent hardware, accessories and supplies against defects in materials and workmanship for the period specified above. If Agilent receives notice of such defects during the warranty period, Agilent will, at its option, either repair or replace products which prove to be defective. Replacement products may be either new or like-new.

2. Agilent warrants that Agilent software will not fail to execute its programming instructions, for the period specified above, due to defects in material and workmanship when properly installed and used. If Agilent receives notice of such defects during the warranty period, Agilent will replace software media which does not execute its programming instructions due to such defects.

3. Agilent does not warrant that the operation of Agilent products will be interrupted or error free. If Agilent is unable, within a reasonable time, to repair or replace any product to a condition as warranted, customer will be entitled to a refund of the purchase price upon prompt return of the product.

4. Agilent products may contain remanufactured parts equivalent to new in performance or may have been subject to incidental use.

5. The warranty period begins on the date of delivery or on the date of installation if installed by Agilent. If customer schedules or delays Agilent installation more than 30 days after delivery, warranty begins on the 31st day from delivery.

6. Warranty does not apply to defects resulting from (a) improper or inadequate maintenance or calibration, (b) software, interfacing, parts or supplies not supplied by Agilent, (c) unauthorized modification or misuse, (d) operation outside of the published environmental specifications for the product, or (e) improper site preparation or maintenance.

7. TO THE EXTENT ALLOWED BY LOCAL LAW, THE ABOVE WARRANTIES ARE EXCLUSIVE AND NO OTHER WARRANTY OR CONDITION, WHETHER WRITTEN OR ORAL, IS EXPRESSED OR IMPLIED AND AGILENT SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE.

8. Agilent will be liable for damage to tangible property per incident up to the greater of $300,000 or the actual amount paid for the product that is the subject of the claim, and for damages for bodily injury or death, to the extent that all such damages are determined by a court of competent jurisdiction to have been directly caused by a defective Agilent product.

9. TO THE EXTENT ALLOWED BY LOCAL LAW, THE REMEDIES IN THIS WARRANTY STATEMENT ARE CUSTOMER'S SOLE AND EXLUSIVE REMEDIES. EXCEPT AS INDICATED ABOVE, IN NO EVENT WILL AGILENT OR ITS SUPPLIERS BE LIABLE FOR LOSS OF DATA OR FOR DIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL (INCLUDING LOST PROFIT OR DATA), OR OTHER DAMAGE, WHETHER BASED IN CONTRACT, TORT, OR OTHERWISE.

FOR CONSUMER TRANSACTIONS IN AUSTRALIA AND NEW ZEALAND: THE WARRANTY TERMS CONTAINED IN THIS STATEMENT, EXCEPT TO THE EXTENT LAWFULLY PERMITTED, DO NOT EXCLUDE, RESTRICT OR MODIFY AND ARE IN ADDITION TO THE MANDATORY STATUTORY RIGHTS APPLICABLE TO THE SALE OF THIS PRODUCT TO YOU.

U.S. Government Restricted Rights

The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227- 7013 (Oct 1988), DFARS 252.211-7015 (May 1991) or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987)(or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the Agilent standard software agreement for the product involved.

Agilent Technologies

E1442A 64-Channel Form C Switch Module User's Manual
Edition 3

Documentation History

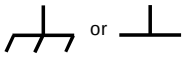
All Editions and Updates of this manual and their creation date are listed below. The first Edition of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct or add additional information 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, 1994
Edition 2 . . . . . March, 1996
Edition 3 . . . . . October, 2000

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.

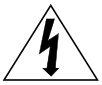
Indicates the field wiring terminal that must be connected to earth ground before operating the equipment — protects against electrical shock in case of fault.



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

Alternating current (AC)

Direct current (DC).



Warning. Risk of electrical shock.

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. Agilent Technologies 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 Agilent 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 Agilent for service and repair to ensure that safety features are maintained.

Agilent Technologies	**DECLARATION OF CONFORMITY** **According to ISO/IEC Guide 22 and CEN/CENELEC EN 45014**	

Manufacturer's Name: Agilent Technologies, Inc.
Manufacturer's Address: *Measurement Products Unit*
815 14th Street S.W.
Loveland, CO 80537 USA

Declares, that the product

Product Name: 64-Channel Form C Switch
Model Number: E1442A
Product Options: *This declaration includes all options of the above product(s).*

Conforms with the following European Directives:

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

Conforms with the following product standards:

	Standard	**Limit**
EMC	*IEC 61326-1:1997 + A1:1998 / EN 61326-1:1997 + A1:1998*	
	CISPR 11:1997 + A1:1997 / EN 55011-1991	*Group 1, Class A [1]*
	IEC 61000-4-2:1995+A1998 / EN 61000-4-2:1995	*4 kV CD, 8 kV AD*
	IEC 61000-4-3:1995 / EN 61000-4-3:1995	*3 V/m, 80-1000 MHz*
	IEC 61000-4-4:1995 / EN 61000-4-4:1995	*0.5 kV signal lines, 1 kV power lines*
	IEC 61000-4-5:1995 / EN 61000-4-5:1995	*0.5 kV line-line, 1 kV line-ground*
	IEC 61000-4-6:1996 / EN 61000-4-6:1996	*3 V, 0.15-80 MHz*
	IEC 61000-4-11:1994 / EN 61000-4-11:1994	*1 cycle, 100%*
	Canada: ICES-001:1998	
	Australia/New Zealand: AS/NZS 2064.1	
Safety	*IEC 61010-1:1990+A1:1992+A2:1995 / EN 61010-1:1993+A2:1995*	
	Canada: CSA C22.2 No. 1010.1:1992	
	UL 3111-1	

Supplemental Information:

[1] The product was tested in a typical configuration with Agilent Technologies test systems.

September 5, 2000
Date

Name

Quality Manager
Title

For further information, please contact your local Agilent Technologies sales office, agent or distributor.
Authorized EU-representative: Agilent Technologies Duetschland GmbH, Herrenberger Strase 130, D 71034 Boblingen, Germany

Revision: A.03 Issue Date: 09/05/00

Notes:

Chapter 1
Getting Started

Using This Chapter

This chapter shows how to get started using the E1442A 64-Channel Form C Switch Module. It gives guidelines to configure, install and program the module. Chapter contents include:

Switch Description

The E1442A 64-Channel Form C Switch Module is a VXIbus C-Size register-based slave device that can operate in a C-Size VXIbus mainframe or in a VMEbus mainframe. The E1442A switch consists of a Form C switch module and one of three types of terminal modules (Standard, Option 010, and Option 020). The terms "Form C Switch" and "switch" refer to the E1442A switch module.

The switch "instrument" is the firmware running in the E1406 Command Module. This firmware is the instrument driver providing Standard Commands for Programmable Instruments (SCPI) programming capability. The term "switchbox" refers to a switch instrument consisting of one or more switch modules.

Programming the E1442A can be done through the command module using SCPI or via direct register access (register-based programming).

Switch Front Panel

The Form C switch consists of a component module and a terminal module. User inputs are connected to the Form C switch NO (Normally Open), NC (Normally Closed), and C (Common) terminal connections on one of the three available terminal modules. Figure 1-1 shows the switch module front panel and the connector pinouts that mate to the terminal module.

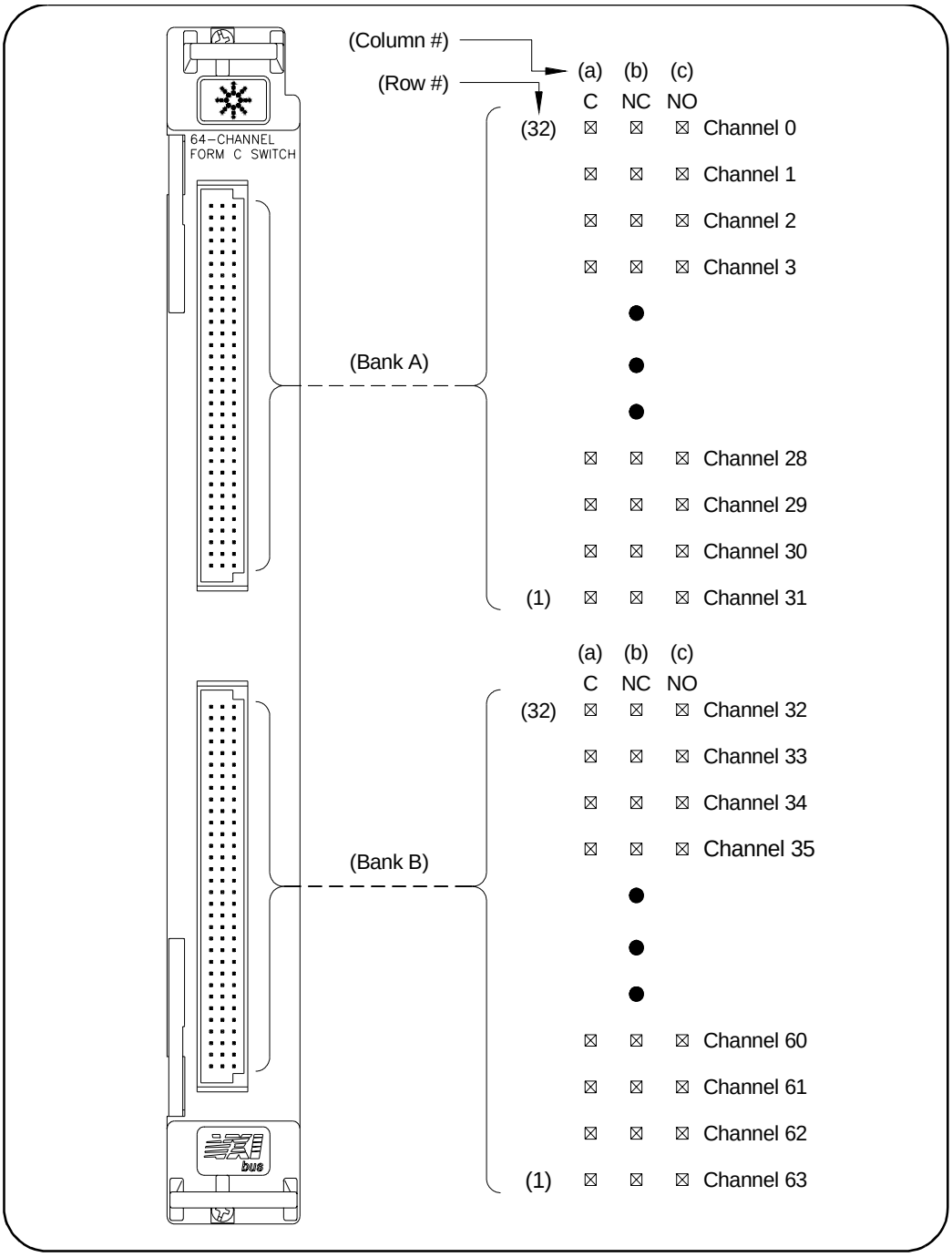


Figure 1-1. E1442A Form C Switch Front Panel

Switch Block Diagram

Figure 1-2 is a simplified block diagram of the Form C switch with internal bus and available terminal modules (Standard, Option 010, and Option 020).

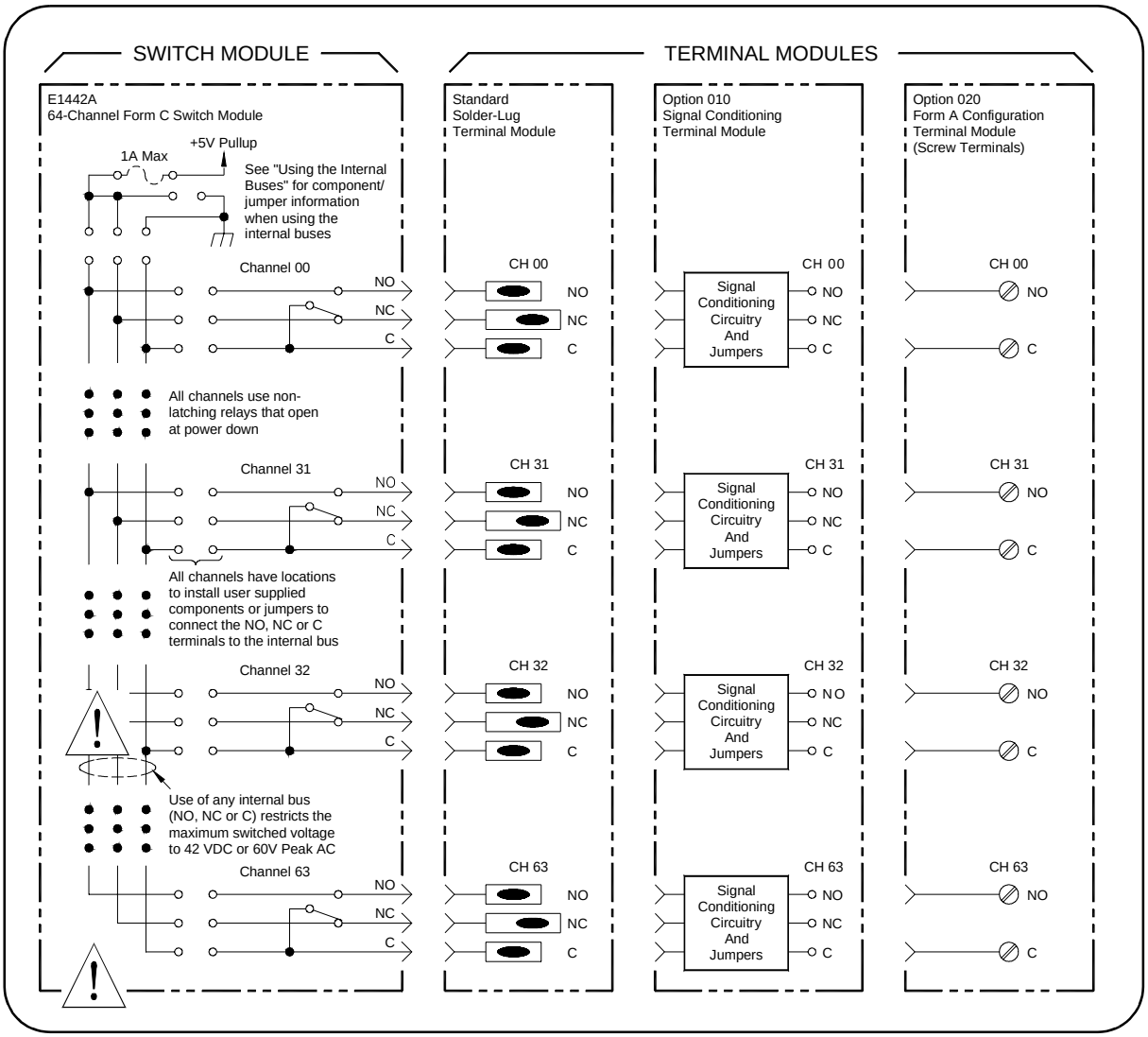


Figure 1-2. E1442A Form C Switch - Simplified Block Diagram

Terminal Module Descriptions

Figure 1-3 shows the Standard Terminal Module Form C configuration with solder lugs, the Option 010 Terminal Module Form C configuration with signal conditioning circuitry, and the Option 020 Form A Screw Terminal configuration.

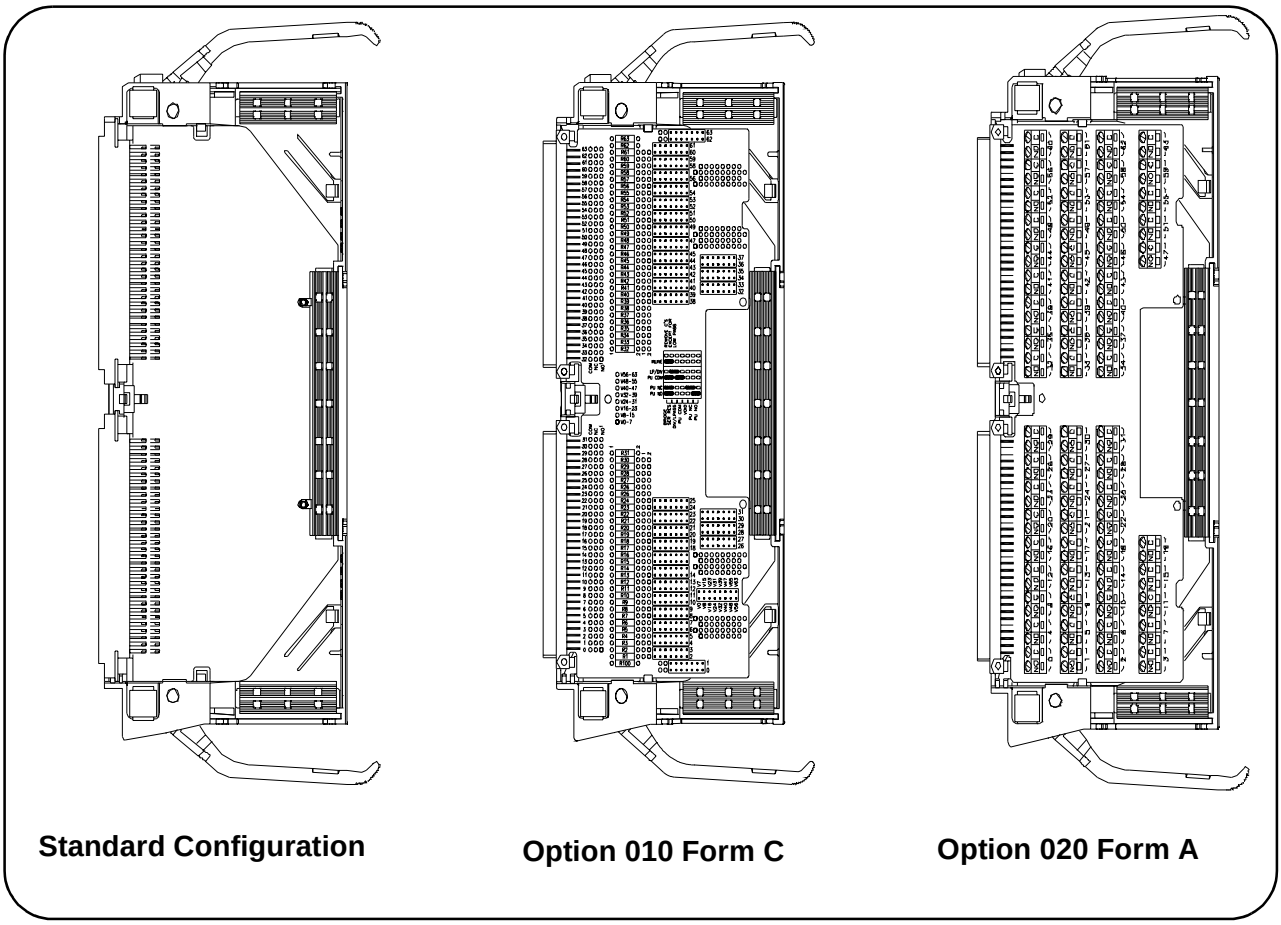


Figure 1-3. Form C Switch - Terminal Modules

Figure 1-4 shows the three terminal modules and options for NO and NC connections for each terminal type. A switch (relay) is **open** when contact is made between the normally closed (NC) contact and common (C). A switch is **closed** when contact is made between the normally open (NO) and common (C). Any combination of open or closed states is allowed at one time for all channels on the module.

	Terminal Module Type		
	Standard Form C Solder Lug	Option 010 Form C Signal Conditioning	Option 020 Form A Screw Terminal
Relay Open	Load 1	Load 1	No Connection
Relay Closed	Load 2	Load 2	Load 2

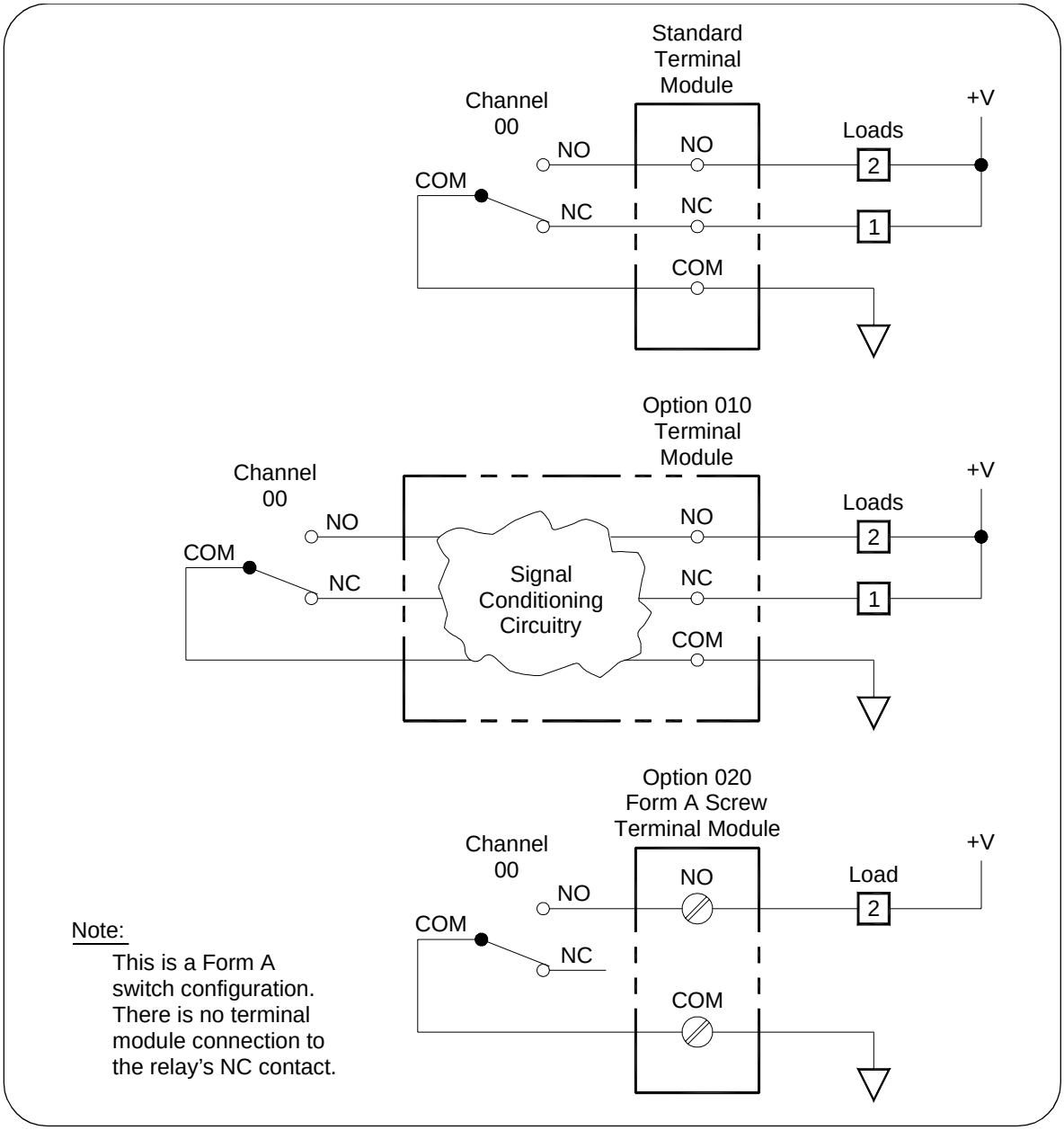


Figure 1-4. Terminal Module Configurations

Configuring the Switch

This section gives guidelines to configure the switch, including the following items. See "Configuring the Terminal Modules" for information on configuring the terminal modules.

- Warnings and Cautions
- Setting the Logical Address
- Setting Interrupt Priority
- Using the Internal Bus
- Installing the Switch in a Mainframe

Warnings and Cautions

You must observe the warnings and cautions that follow in addition to the general warnings and cautions in the front matter in this manual when installing, configuring, or removing the module.

WARNING **SHOCK HAZARD. Only qualified, service-trained personnel aware of the hazards involved should install, configure, or remove the module. Disconnect all power sources from the mainframe, the terminal module and installed modules before installing or removing a module.**

WARNING **SHOCK HAZARD. When handling user wiring connected to the terminal module, consider the highest voltage present accessible on any terminal.**

WARNING **SHOCK HAZARD. Use wire with an insulation rating greater than the highest voltage which will be present on the terminal module. Do not touch any circuit element connected to the terminal module if any other connector to the terminal module is energized to more than 30 VAC RMS or 60 VDC.**

CAUTION **MAXIMUM VOLTAGE/CURRENT. Maximum allowable voltage per channel terminal-to-terminal or terminal-to-chassis for the switch module is 150 Vdc or 150 Vac RMS (210 Vac peak). Maximum current per channel is 1 Adc or 1 Aac RMS (non-inductive). Maximum transient voltage is 1300V peak. Maximum power input is 40 Wdc or 40 VA per channel, 320 Wdc or 320 VA per module. Exceeding any limit or use outside the parameters specified in Appendix A and by these warnings and cautions may damage the switch module and impair the protection provided by the module.**

CAUTION **WIRING TERMINAL MODULE: When wiring to the terminal connectors on a terminal module, do not exceed a 5mm strip back of insulation to prevent the possibility of shorting to other wiring on adjacent terminals.**

CAUTION **STATIC-SENSITIVE DEVICE. Use anti-static procedures when removing, configuring, cleaning and installing a module. Since the switch module is susceptible to static discharges, do not install the module without its metal**

CAUTION **CLEANING THE FRONT PANEL. Disconnect power from the mainframe and remove the module to be cleaned. Clean the front panel with a soft cloth dampened either in clean water or in water containing a mild detergent. Do not use abrasive cleaners. Do not use an excessively wet cloth or allow excessive water to migrate inside the module. Let the panel dry thoroughly before reinstalling the module.**

Setting the Logical Address

The E1442A switch module logical address is set with the Logical Address Switch (LADDR) on the module. The factory setting for the LADDR is 120. Valid addresses are from 1 to 254. The module logical address value is set by the sum of the decimal values of the switches that are CLOSED.

Example: Setting a LADDR

For example, in Figure 1-5, switches 3, 4, 5, and 6 are CLOSED. Since the decimal value of switch 3 = 8, the value of switch 4 = 16, the value of switch 5 = 32, and the value of switch 6 = 64, the LADDR set = 8 + 16 + 32 + 64 = 120.

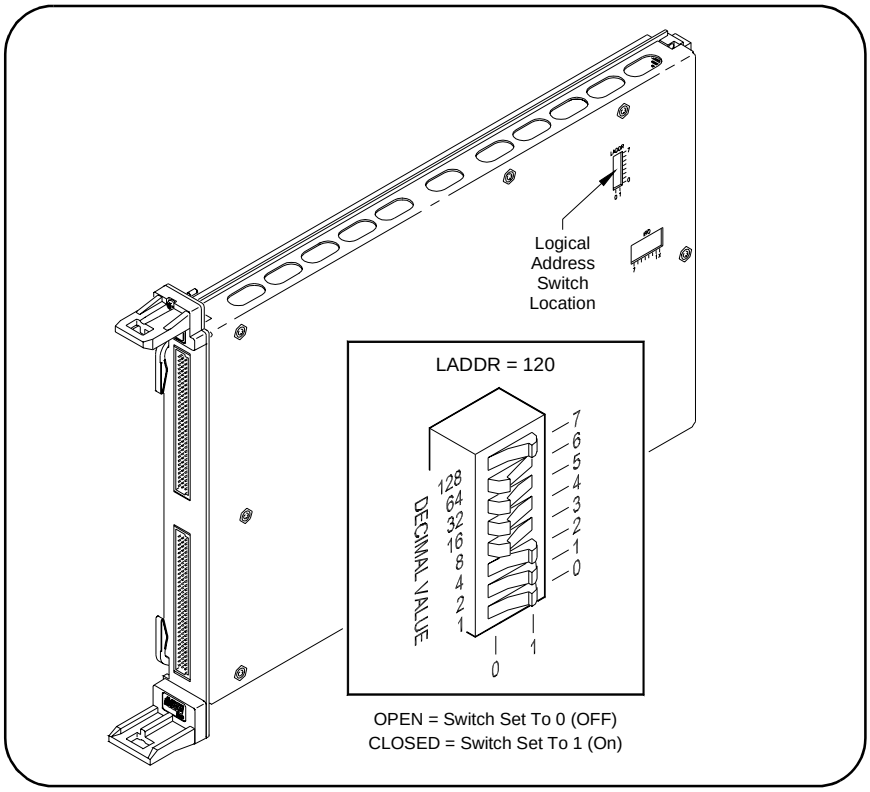


Figure 1-5. Setting the Logical Address (LADDR)

Determining the LADDR

To determine the logical address switch (LADDR) setting for your application, you must first decide whether the switch is to be used as a single-module switchbox or as a multiple-module switchbox. When using an E1406 Command Module, the LADDR value must be a multiple of 8 if the module is the first module in a switchbox used with a VXIbus command module using SCPI commands.

- **Single-module switchbox.** The module must be addressed so it can be recognized as an instrument, such as 48, 56, etc..
- **Multiple-module switchbox.** In this configuration, two or more modules form the switchbox. The first module must be addressed so it can be recognized as an instrument and the other modules in the group have addresses sequentially following the first module, such as 120, 121, 122 ....

Figure 1-6 shows some examples of single- and multiple-module switchbox arrangements.For the multiple-module switchbox (top figure), the channel address (*channel_list*) has the form (@*ccnn*) where *cc* = card number and *nn* = channel number. For example, channel 45 on card number 02 is addressed by (@245).

The multiple- and single-module switchbox (bottom figure), has two switchboxes: a multiple-module switchbox at logical address 120 and a single-module switchbox at address 48. The single-module switchbox has channel addresses of the form (@1*nn*). Its card number is 1.

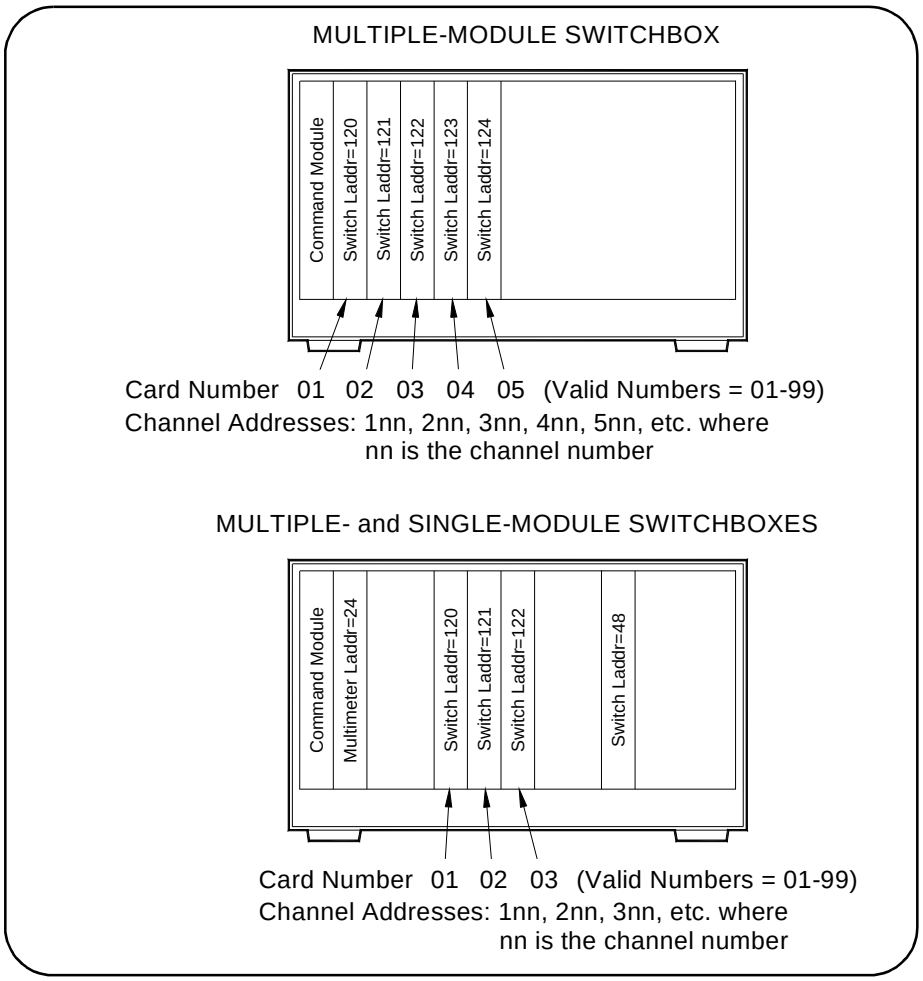


Figure 1-6. Typical Switchbox Arrangements

Setting Interrupt Priority

Interrupts are enabled at power-up, after a SYSRESET, or after resetting the module via the control register. An interrupt is generated after any channel enable register is accessed when interrupts are enabled. The interrupt is generated approximately 13 ms after one of the registers is accessed.

The interrupt priority jumper selects which priority level will be asserted. The interrupt priority jumper is set in position 1 as shipped from the factory. For most applications this priority level should not have to be changed. The interrupts are disabled when set to level X. The interrupt priority jumpers are identified on the sheet metal shield. A hole has been cut into it for access. Interrupts can also be disabled using the Control Register. See Figure 1-7 for Interrupt Request Level Jumper locations.

To change the setting, remove the jumper or jumpers from their current position and place on the level you desire. If the card uses two 2-pin jumpers, both jumpers must be placed in the same row for proper operation. See the applicable mainframe manual to make sure backplane jumpers are configured correctly.

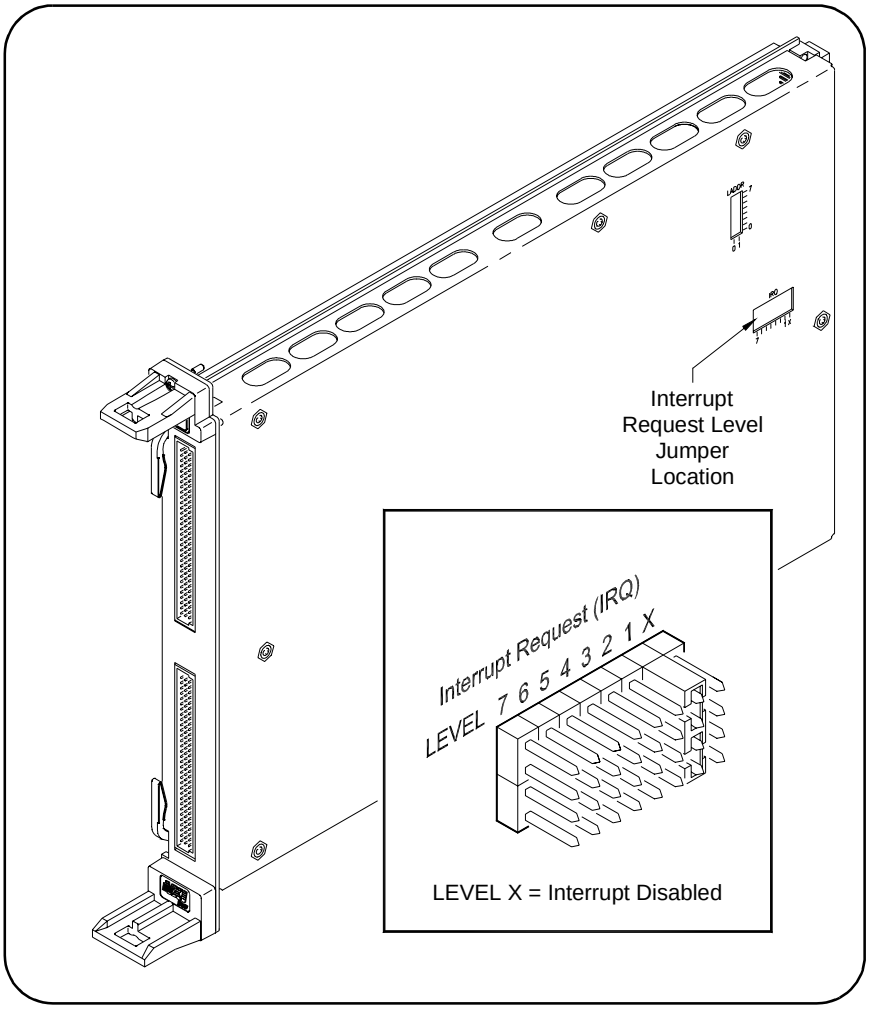


Figure 1-7. Setting Interrupt Request (IRQ) Priority

Using the Internal Buses

The E1442A 64-Channel Form C Switch Module contains internal buses to which you can connect any channel contact. Figure 1-8 shows channels 0 and 63 and the internal bus structure. There is a bus for the common (C), the normally closed (NC), and the normally open (NO) contacts.

Other jumpers provide the means to connect the NC and NO contacts to a fused +5V pull-up voltage, or to be connected as pull-downs to ground. The common can be connected to ground. Figure 1-9 shows component/jumper locations on the module.

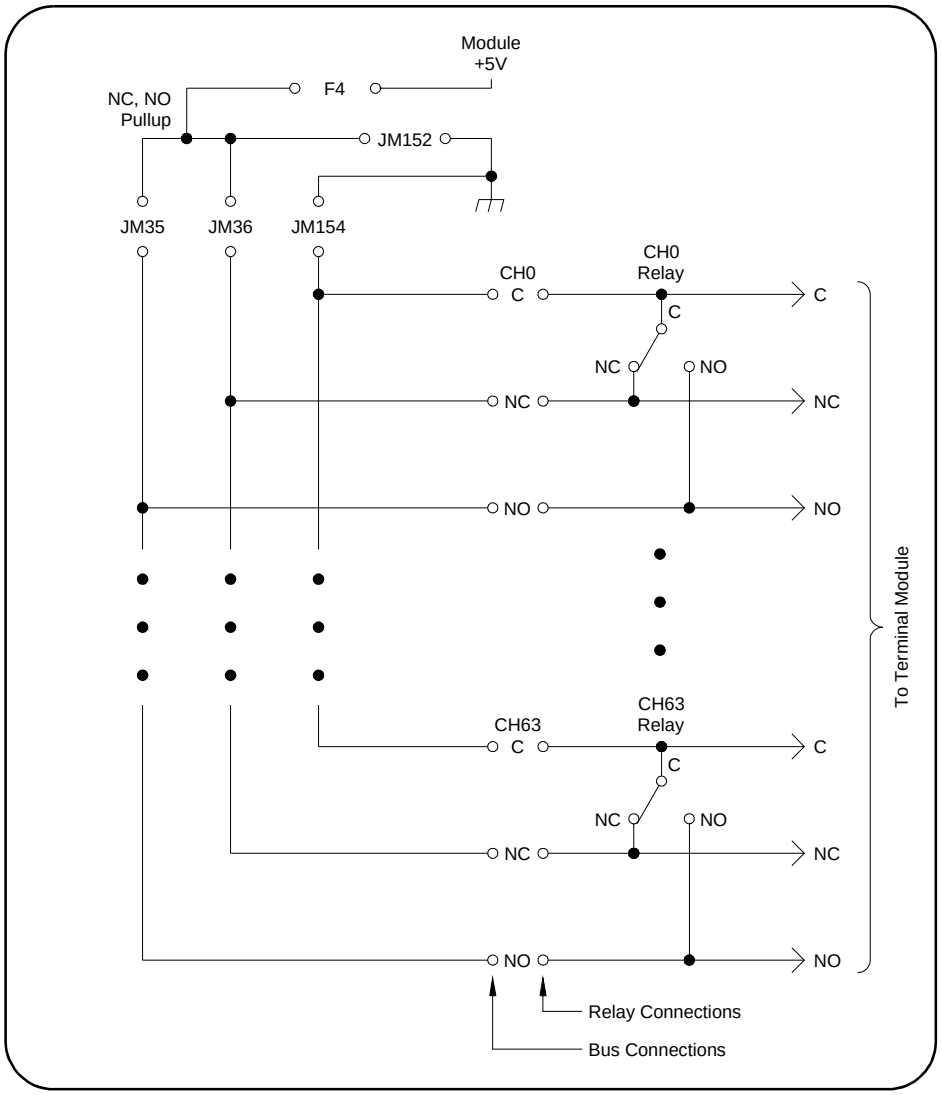


Figure 1-8. Internal Bus Structure

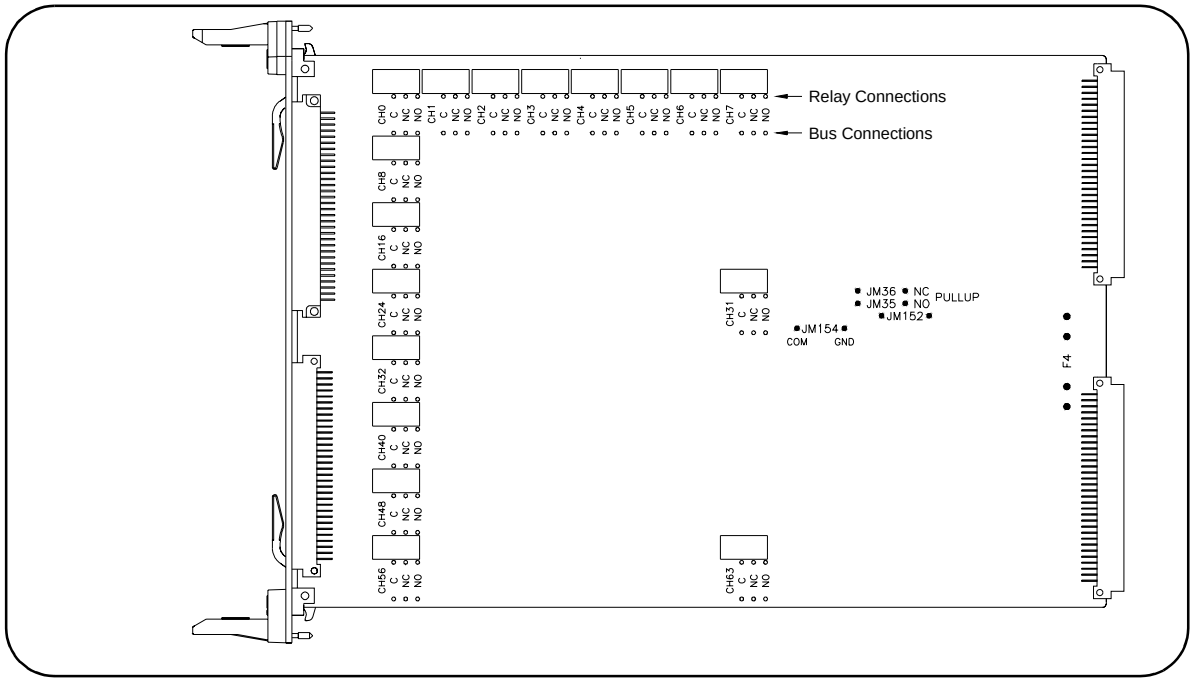


Figure 1-9. Internal Bus Component/Jumper Locations

Installing the Switch in a Mainframe

The E1442A switch module can be installed in any slot (except Slot 0) of a C-size VXIbus mainframe. See Figure 1-10 for installation steps.

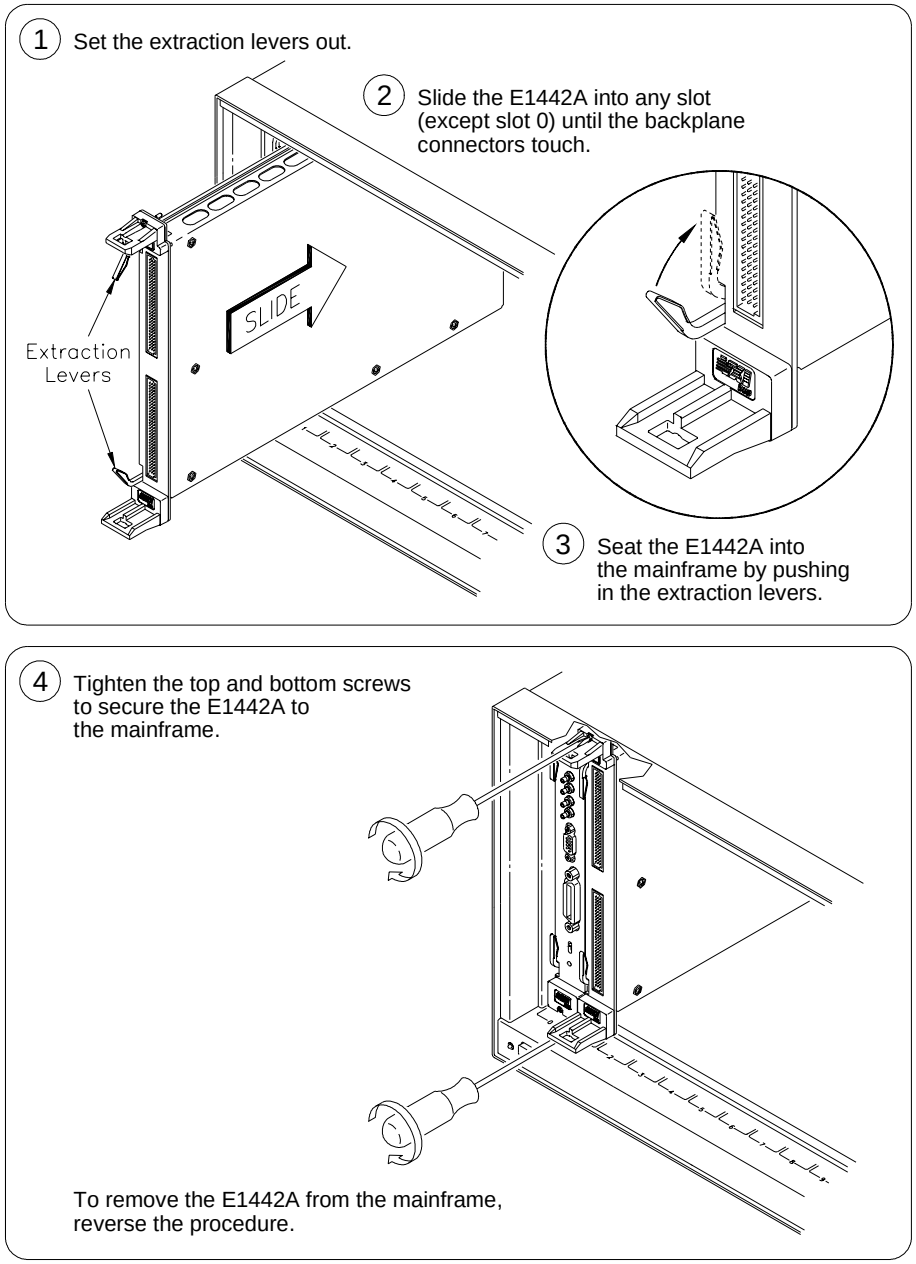


Figure 1-10. Installing the Switch in a VXI Mainframe

Configuring the Terminal Modules

This section gives guidelines to configure the Standard Form C Configuration, Option 010 Form C Configuration, and the Option 020 Form A Configuration terminal modules, including:

- Wiring the Terminal Modules
- Attaching Terminal Modules to the Switch Module
- Configuring the Option 010 Terminal Module

Wiring the Terminal Modules

Figure 1-11 and Figure 1-12 show steps to wire terminal module s. Maximum terminal wire size is No. 16 AWG. Wire ends should be stripped 5mm (0.2 in.) and tinned. When wiring all channels, use a smaller gauge wire (No. 20-22 AWG).

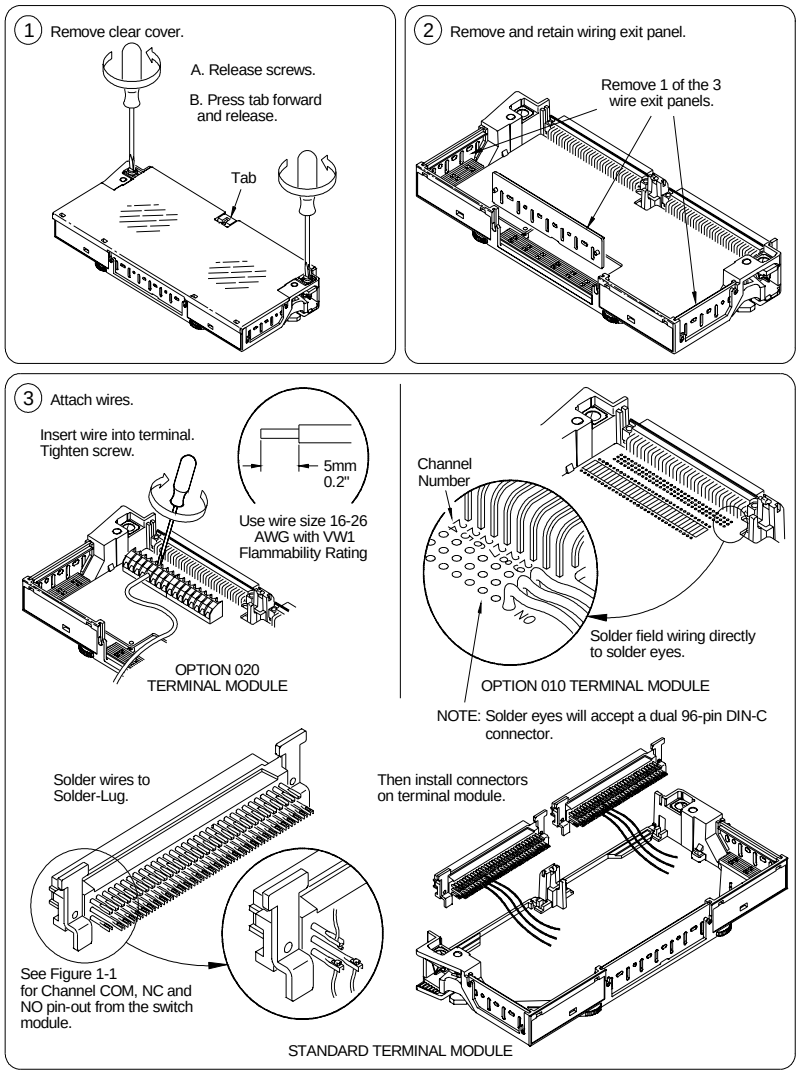


Figure 1-11. Wiring the Terminal Modules (cont'd on Figure 1-12)

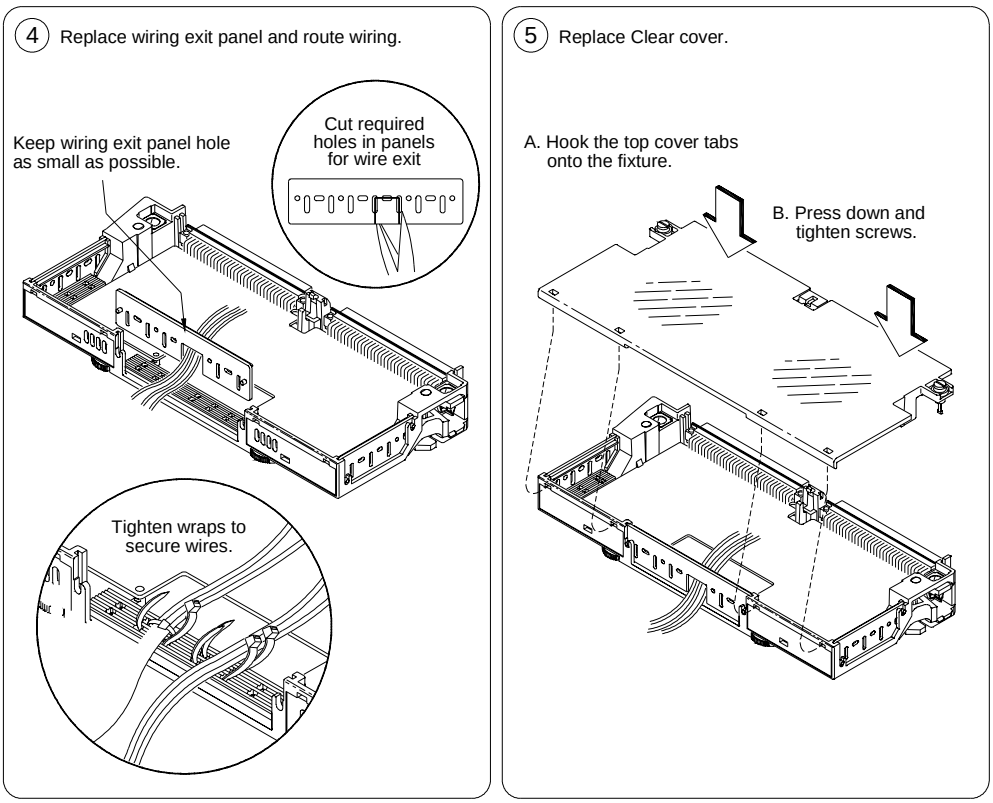


Figure 1-12. Wiring the Terminal Modules (cont'd from Figure 1-11)

Attaching Terminal Modules to the Switch Module

See Figure 1-13 for steps to attach a terminal module to the switch module.

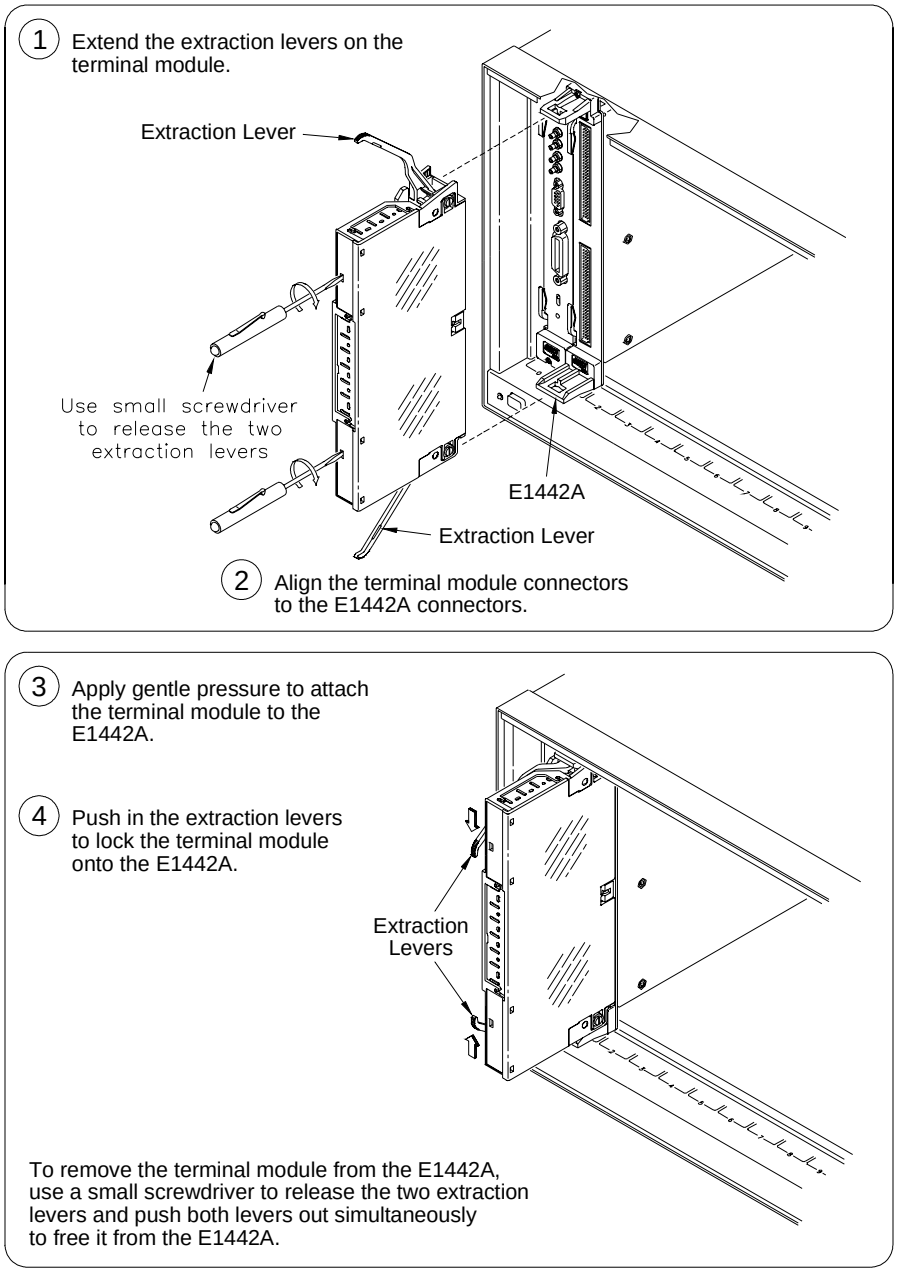


Figure 1-13. Attaching a Terminal Module to the Switch Module

Configuring the Option 010 Terminal Module

This section describes the Option 010 Terminal Module. With this terminal module, you can add components to configure a variety of passive signal conditioning circuits including pullups, pulldowns, and single-ended and differential resistive dividers and filters. User inputs are connected to the module by soldering wires or components to the terminal module PC board.

Terminal Module User Connections

Figure 1-14 shows channels 0 and 1 and associated component and voltage connections (resistors, capacitors, jumpers and voltages). Note the correlation of R0/C0 and R1/C1 with channels 0 and 1 respectively and the associated voltage node V0-7 and user-supplied resistor SIP. Figure 1-15 shows the locations of items on the terminal module.

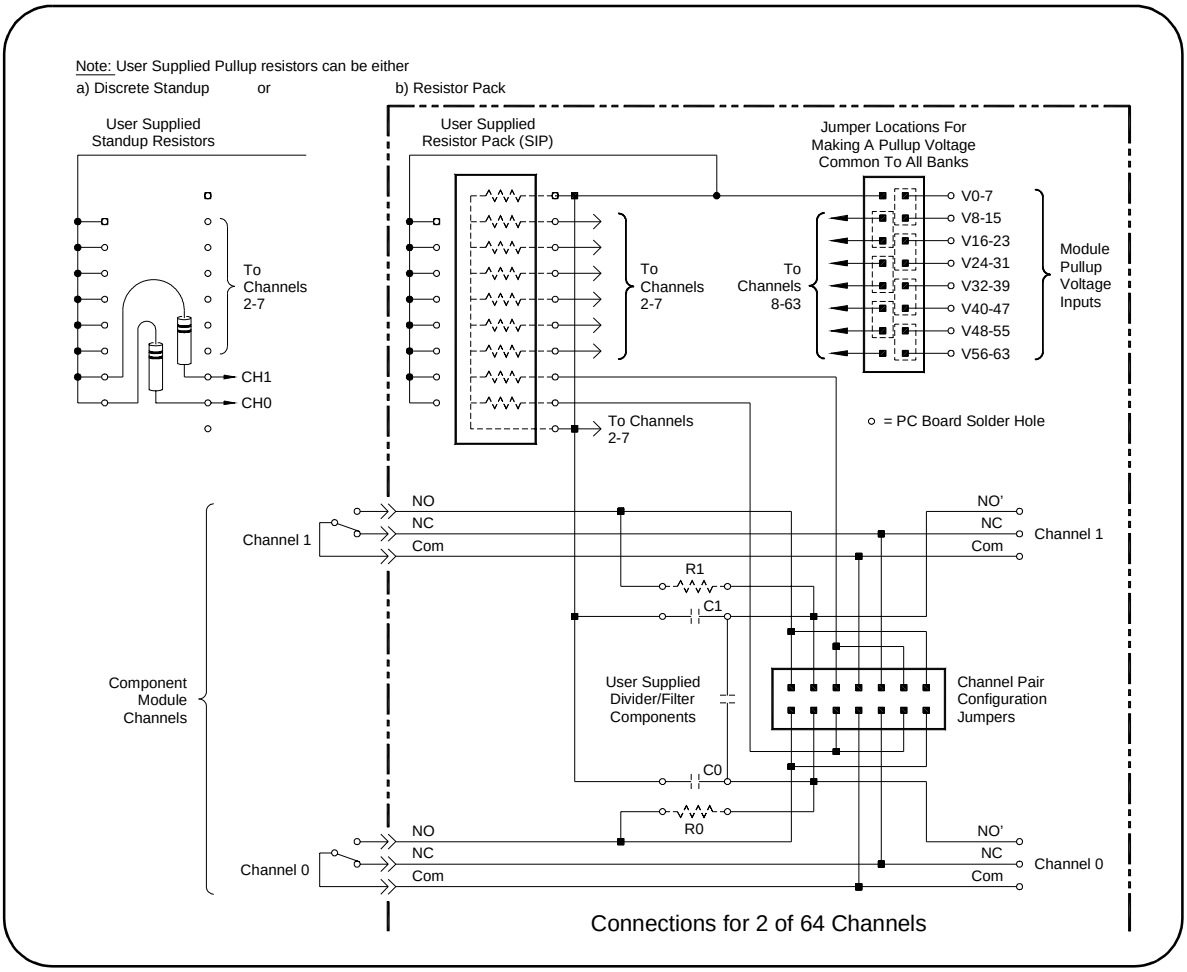


Figure 1-14. Option 010 Terminal Module User Connections

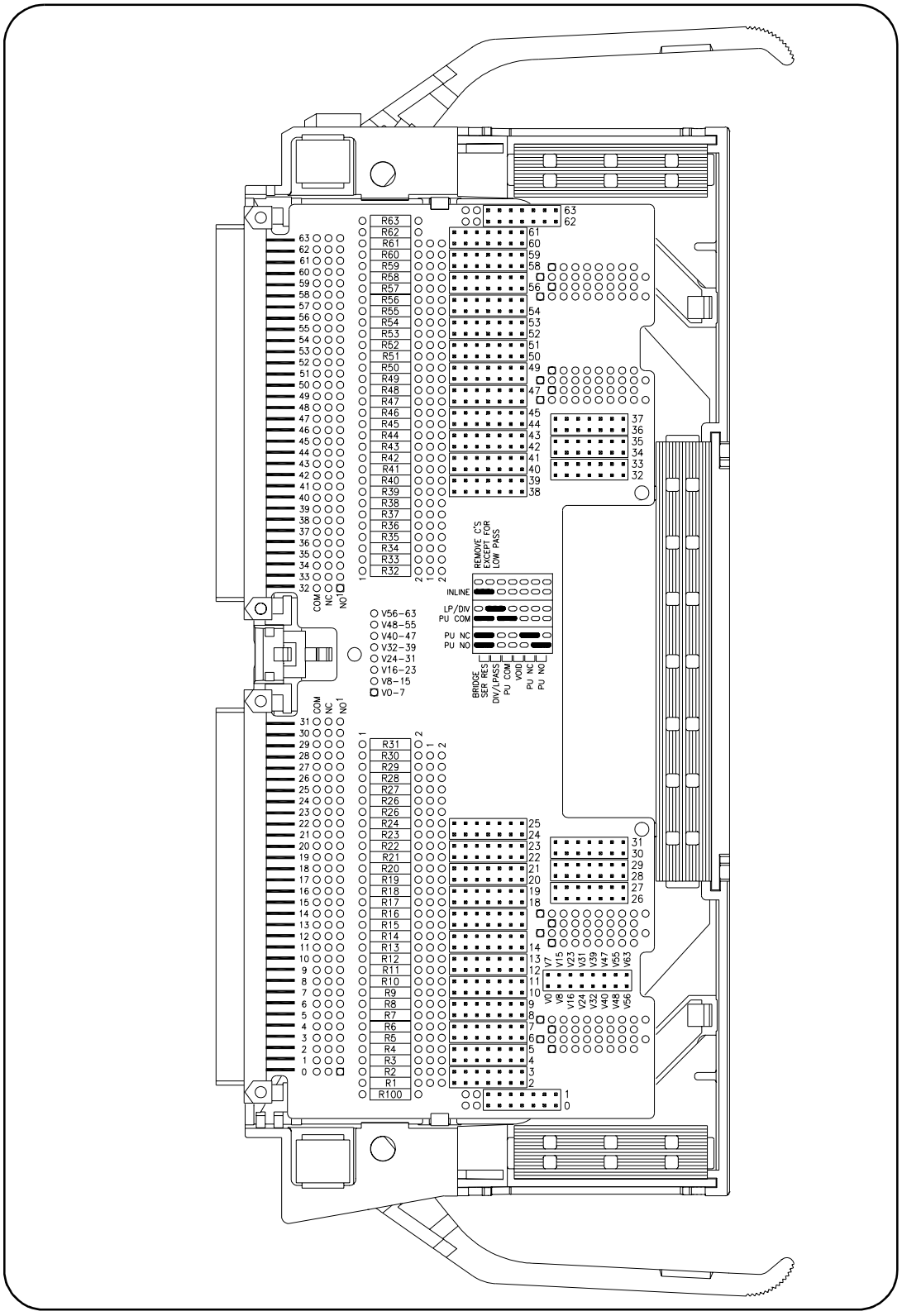


Figure 1-15. Option 010 Terminal Module

Example: Straight-Through Configuration

Any channel of the terminal module can be configured as a straight-through Form C relay. In this mode no resistors or capacitors are included. A two-position jumper is placed on the mode selection jumper. Figure 1-16 shows a typical straight-through configuration. No components are added. Set one configuration jumper as shown in Figure 1-16 (INLINE).

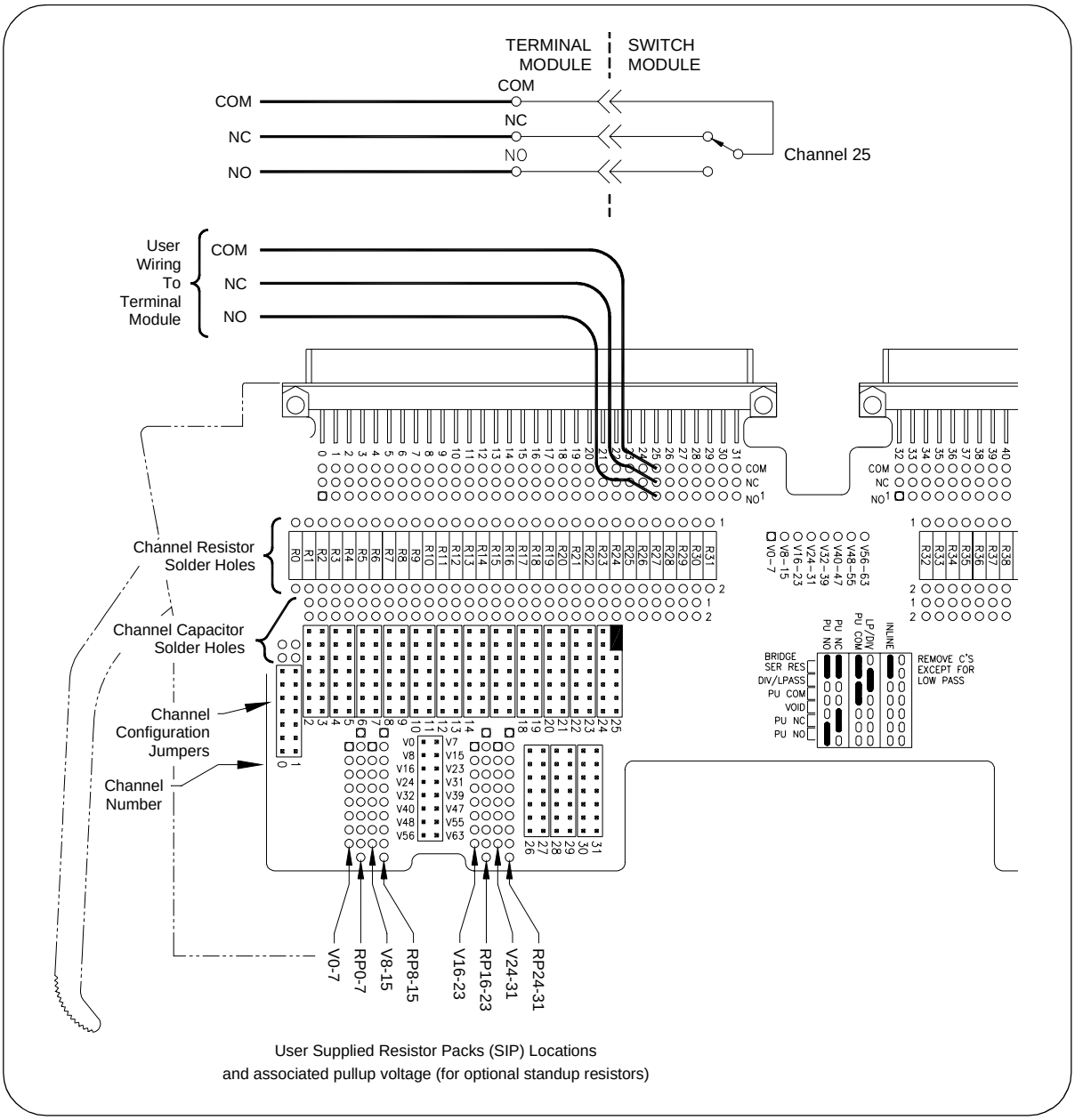


Figure 1-16. Example: Straight-Through Configuration

Example: Resistor Divider Configuration

Any channel can be configured as a resistor divider connected to the normally open (NO) contact of the Form C relay. The user-supplied SIP resistor can be replaced by a standing resistor with it inserted in the solder hole of the SIP and a solder hole directly across from it. The row of solder holes is connected to V24-31.

Figure 1-17 shows the voltage solder holes and identifies the voltage to which the row is connected. For this example, resistor R25 and SIP resistor pack RP24-31 are to be added. Set one configuration jumper as shown in Figure 1-17 (LP/DIV).

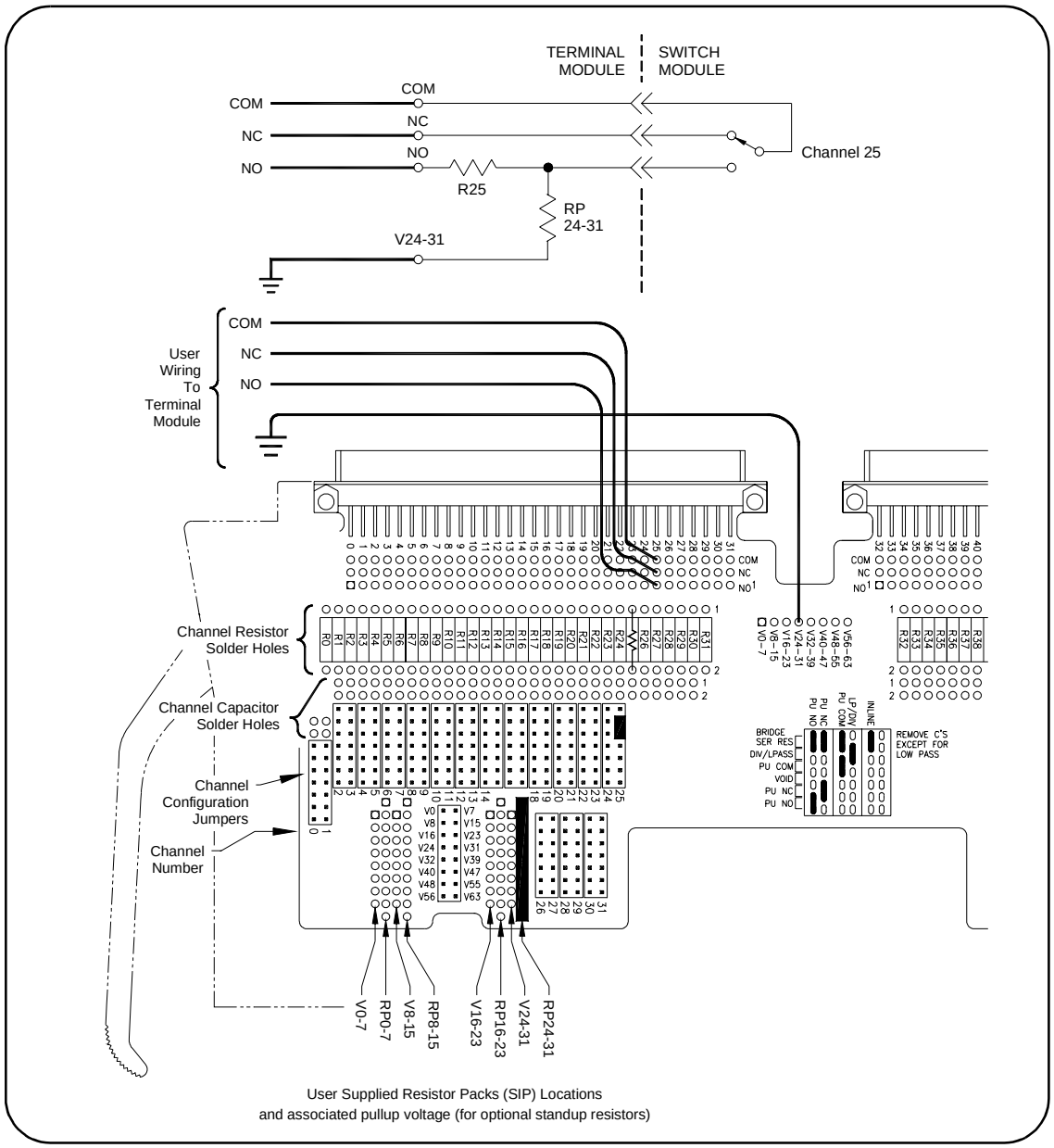


Figure 1-17. Example: Resistor Divider Configuration

Example: Low-Pass Filter Configuration

Any channel can be configured as a low-pass filter connected to the normally open contact of the Form C relay. Figure 1-18 shows a typical low-pass filter configuration. For this example, resistor R25 and capacitor C25 are to be added. No configuration jumpers are required.

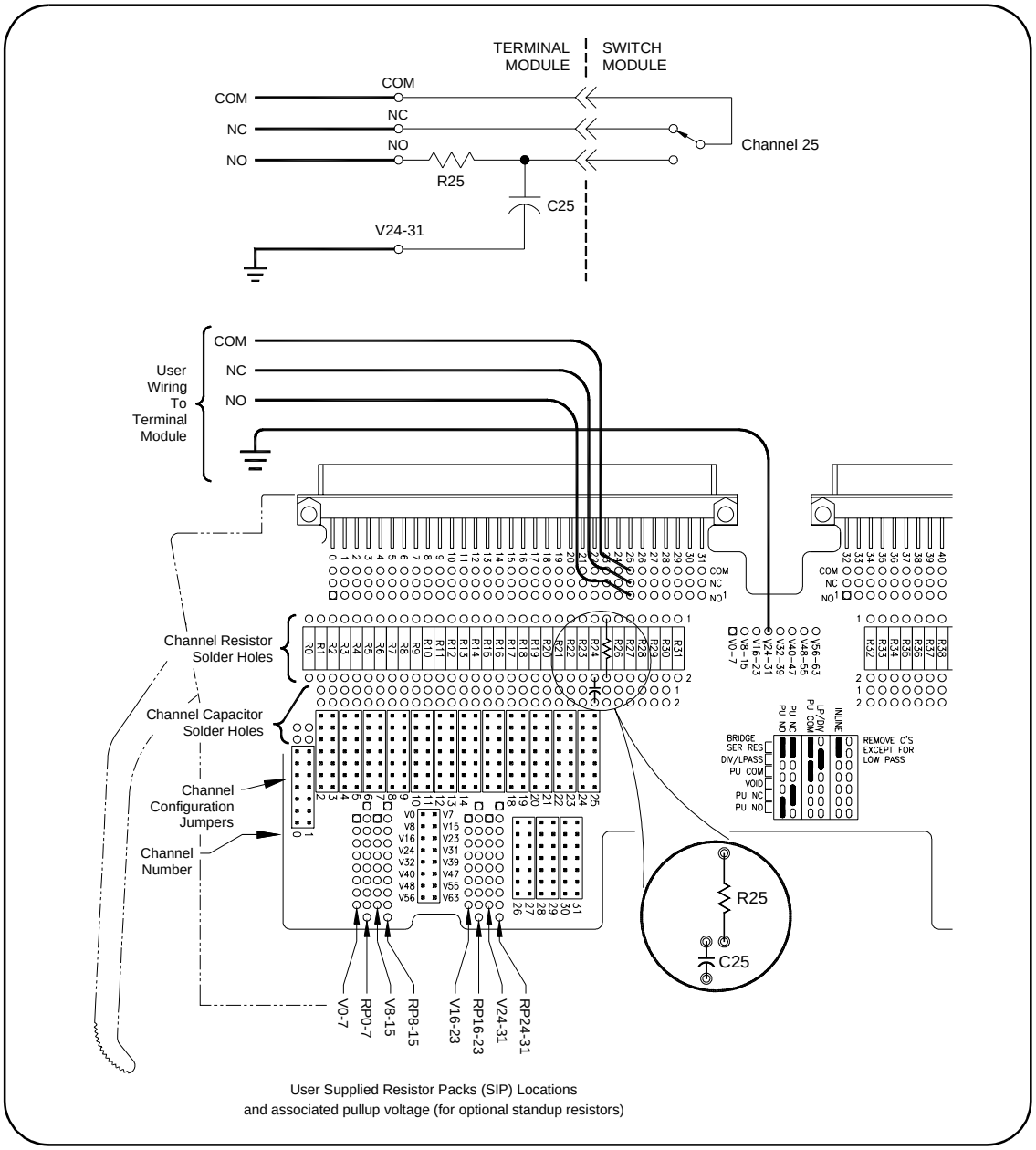


Figure 1-18. Example: Low-Pass Filter Configuration

Example: Common Terminal Pullup Configuration

Any channel can be configured as a pullup (or pulldown) resistor connected to any of the contacts of the Form C relay. Figure 1-19 shows a typical channel 25 with the pullup attached to the COM contact. For this example, the SIP resistor pack RP24-31 is to be added. Set two configuration jumpers as shown in Figure 1-19 (PU COM).

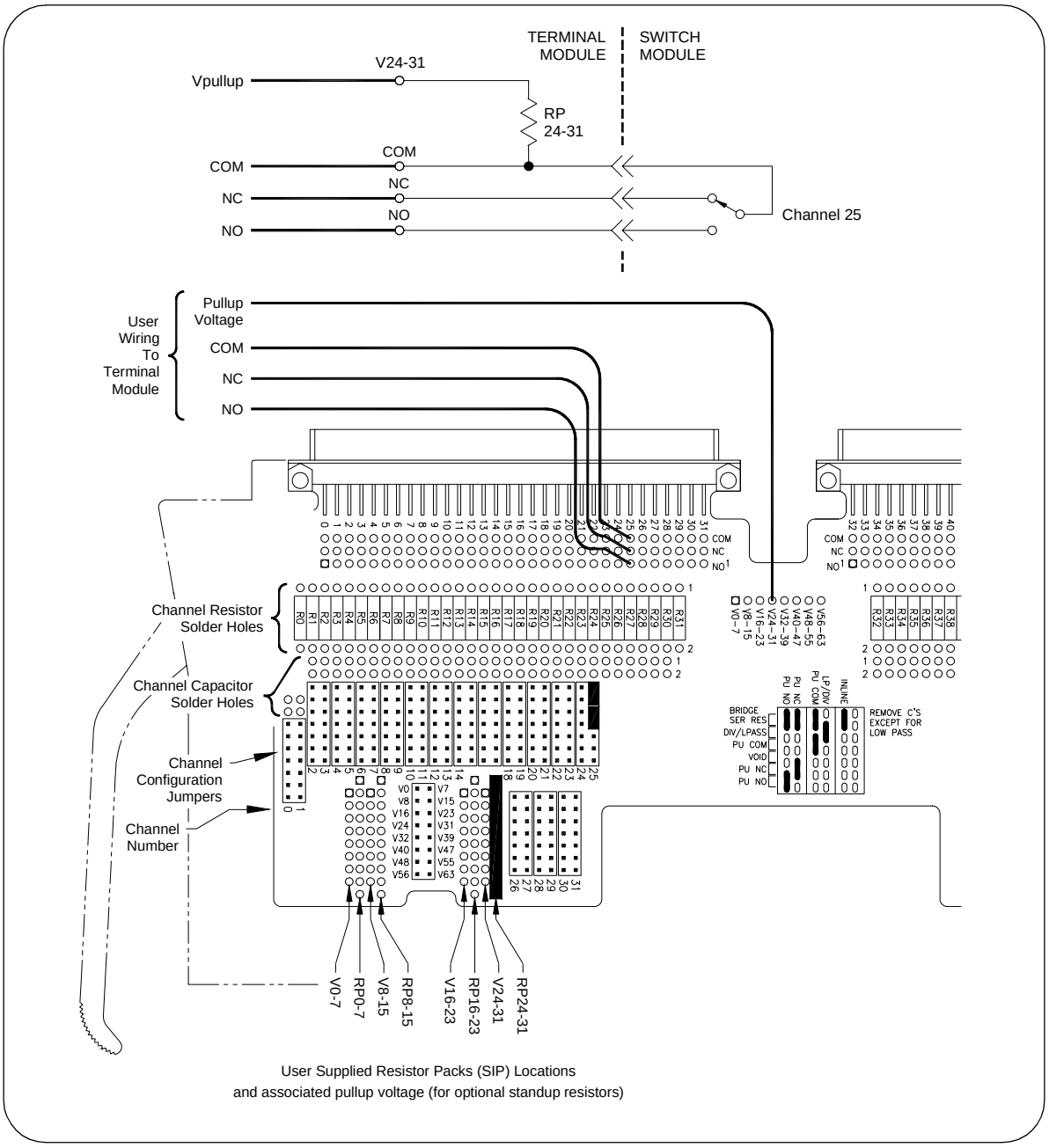
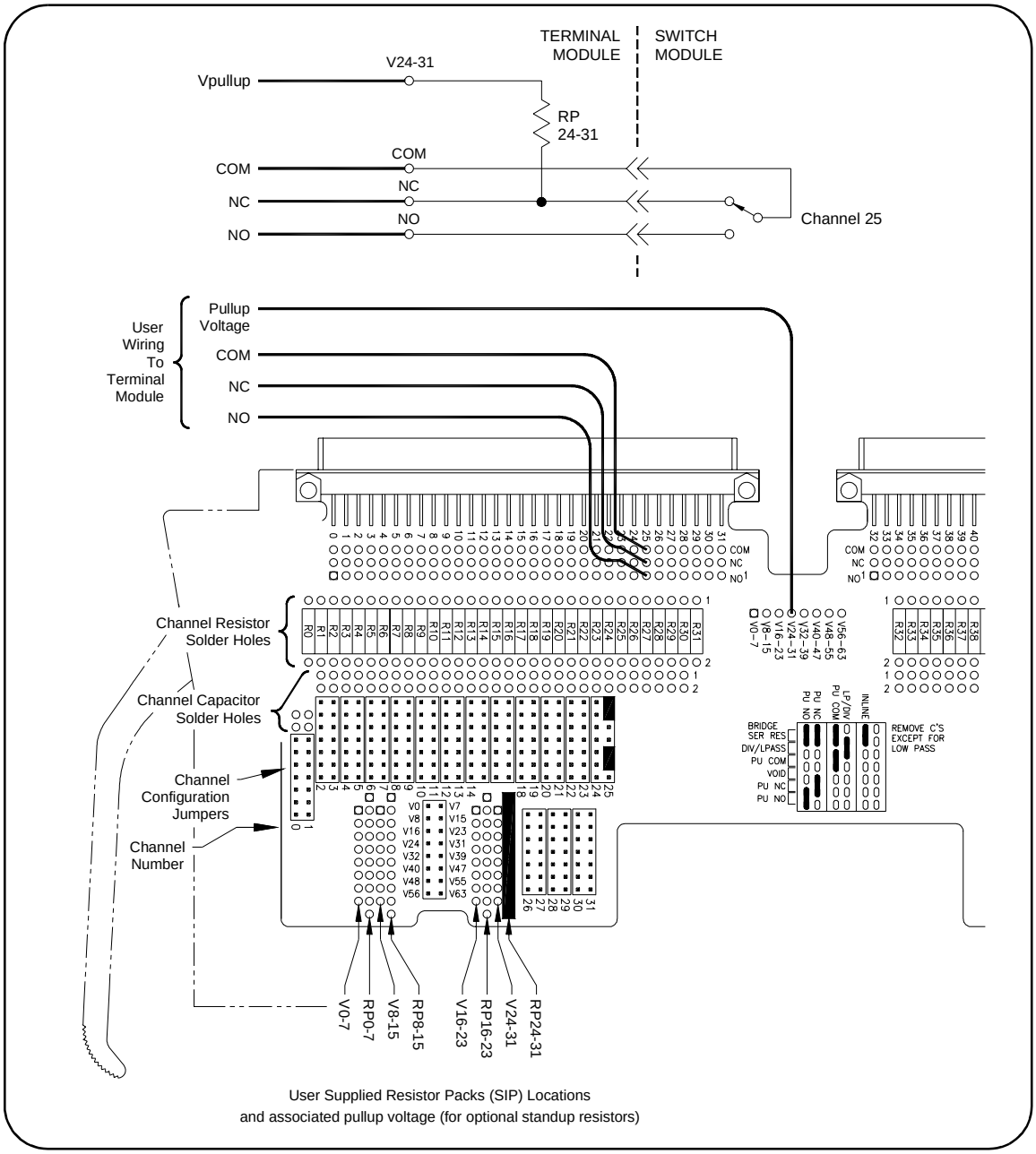


Figure 1-19. Example: Common Terminal Pullup Configuration

Example: Normally Closed Terminal Pullup Configuration

Any channel can be configured as a pullup (or pulldown) resistor connected to any of the contacts of the Form C relay. Figure 1-20 shows channel 25 with the pullup attached to the NC contact. For this example, SIP resistor pack RP24-31 is to be added. Set two configuration jumpers as shown in Figure 1-20 (PU NC).



User Supplied Resistor Packs (SIP) Locations
and associated pullup voltage (for optional standup resistors)

Figure 1-20. Example: Normally Closed Terminal Pullup Configuration

Example: Normally Open Terminal Pullup Configuration

Any channel can be configured as a pullup (or pulldown) resistor connected to any of the contacts of the Form C relay. Figure 1-21 shows channel 25 with the pullup attached to the NO contact. For this example, SIP resistor pack RP24-31 is to be added. Set two configuration jumpers as shown in Figure 1-21 (PU NO).

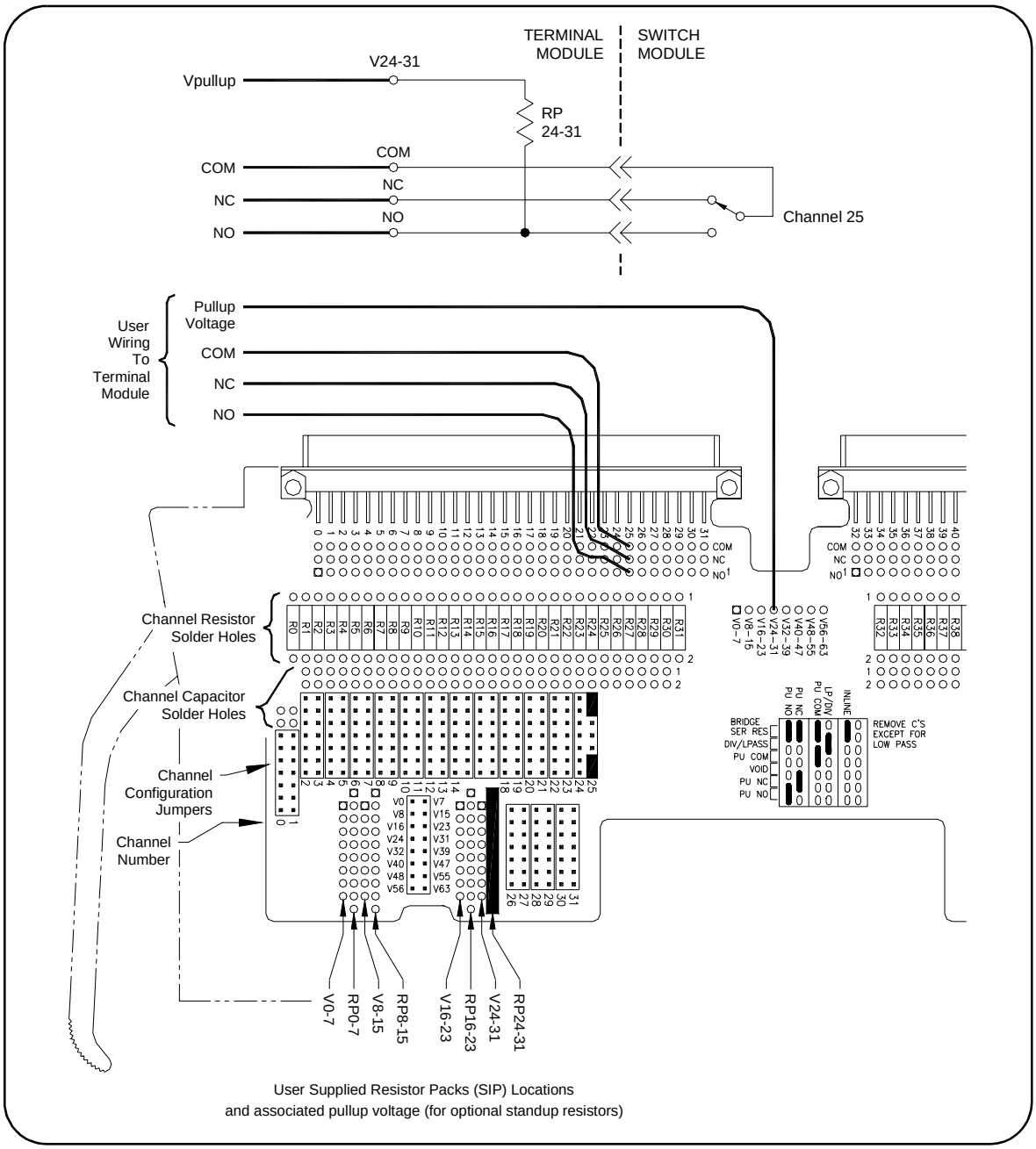


Figure 1-21. Example: Normally Open Terminal Pullup Configuration

Example: Divider with Filter Configuration

Any channel can be configured as a resistor divider with a low-pass filter connected to the normally open contact of the Form C relay. Figure 1-22 shows a typical divider with filter configuration. For this example, resistor R25, capacitor C25, and SIP resistor pack R24-31 are to be added. Set one configuration jumper as shown in Figure 1-22 (LP/DIV).

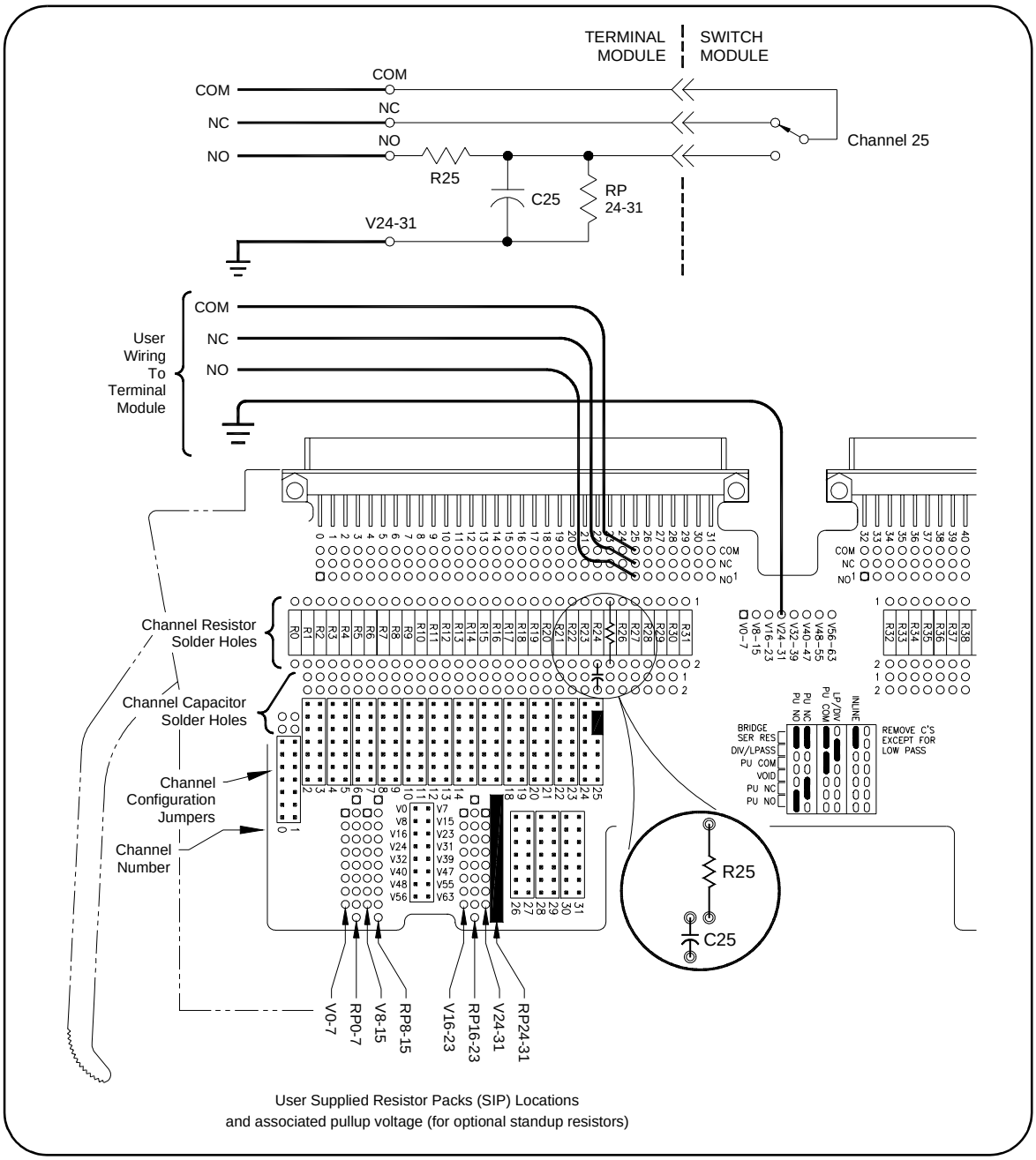


Figure 1-22. Example: Divider with Filter Configuration

Example: Differential Divider or Filter Configuration

Any channel can be configured as a differential divider (with optional filter) connected to the normally open contact of the Form C relay. The differential divider requires that two channels be used.

Figure 1-23 shows channel 24 and 25 in this configuration with the optional filter. For resistors R24 and R25, add a cross-channel capacitor for a differential filter or add a cross-channel resistor for a differential divider. No configuration jumpers are required.

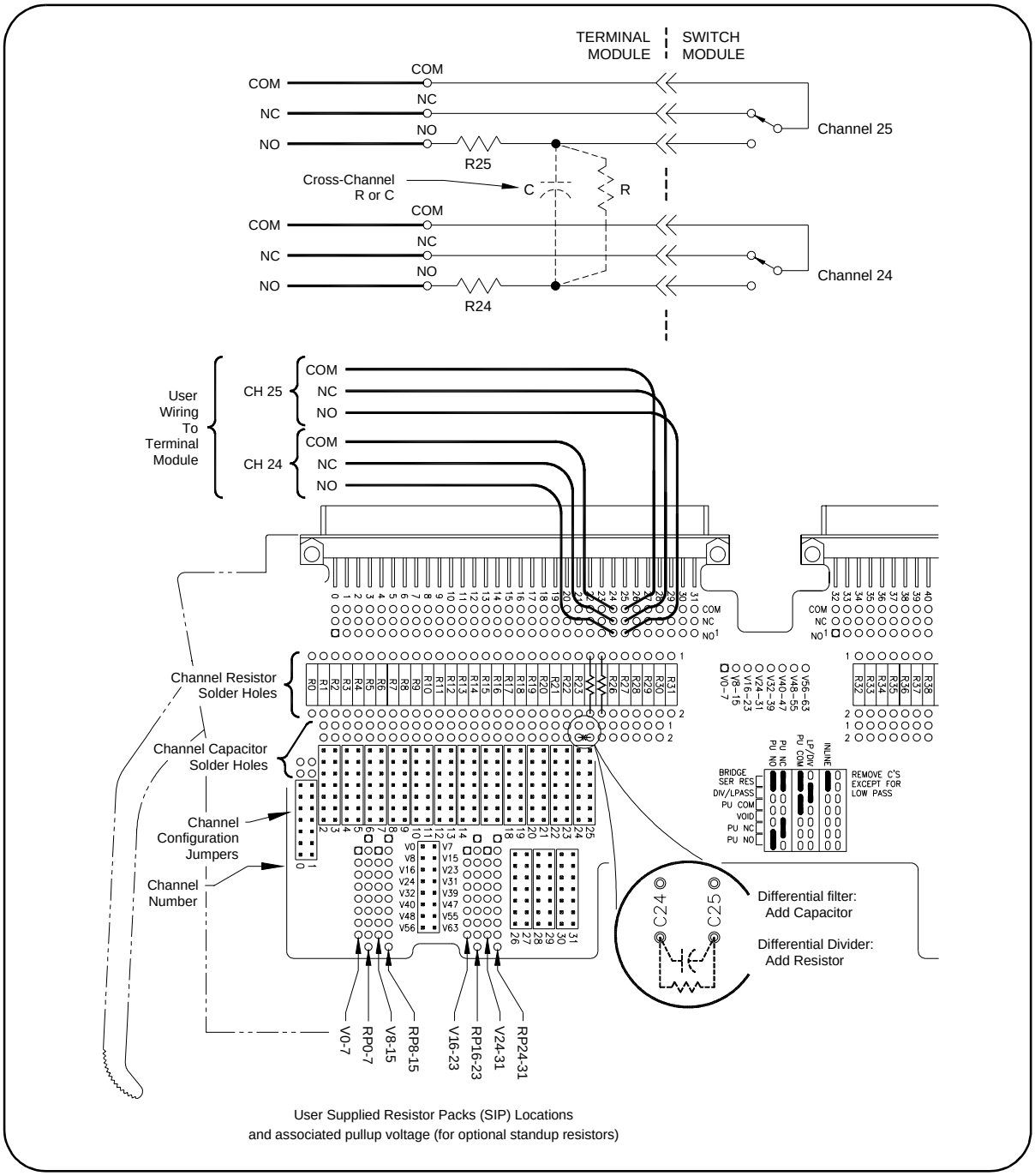


Figure 1-23. Example: Differential Divider or Filter Configuration

Programming the Switch

This section gives guidelines and examples to program the E1442A 64-Channel Form C switch module using Standard Commands for Programmable Instruments (SCPI), including:

- Specifying SCPI Commands
- Start-up Exercises

Specifying SCPI Commands

To program the E1442A switch using SCPI, you must select the computer language, interface address, and SCPI commands to be used. Guidelines to select SCPI commands for the switch follow.

NOTE *This discussion applies only to SCPI programming using the switchbox driver version provided with this module. See Appendix B for information on register-based programming of switch registers.*

To address specific channels within a switch, you must specify the SCPI command and switch channel address. For the Form C switch, use CLOSe <*channel_list*> to connect the normally open (NO) terminal to the common (C) terminal for the channels specified.

Use OPEN <*channel_list*> to connect the normally closed (NC) terminal to the common (C) terminal for the channels specified. Use SCAN <*channel_list*> to close the set of channels specified, one channel at a time.

The Normally Open (NO) contact of each Form C relay is "open" and the Normally Closed (NC) contact of each Form C relay is "closed" when the switch is deactivated (the Common terminal (C) is connected to NC at power-on, after reset or after an open command).

Card Numbers

The switch card number depends on the switchbox configuration (single-module or multiple-module) set for the switches. Leading zeroes can be ignored for the card number. See "Setting Logical Address" in this chapter for more information on setting logical addresses and switchbox configurations.

For a single-module switchbox, the card number is always 01. For a multiple-module switchbox, the card numbers are 01, 02,...,nn. The module with the lowest logical address is card number 01, the module with the next lowest logical address is card number 02, etc.

For example, assume three Form C switches are configured to form a multiple-module switchbox instrument with logical addresses of 120, 121, and 122 as shown in Figure 1-24. Since card number 01 is assigned to the module with the lowest logical address, card number 01 is assigned to the card at logical address 120. Card number 02 is assigned to the card at address 121 and card number 03 is assigned to the card at address 122.

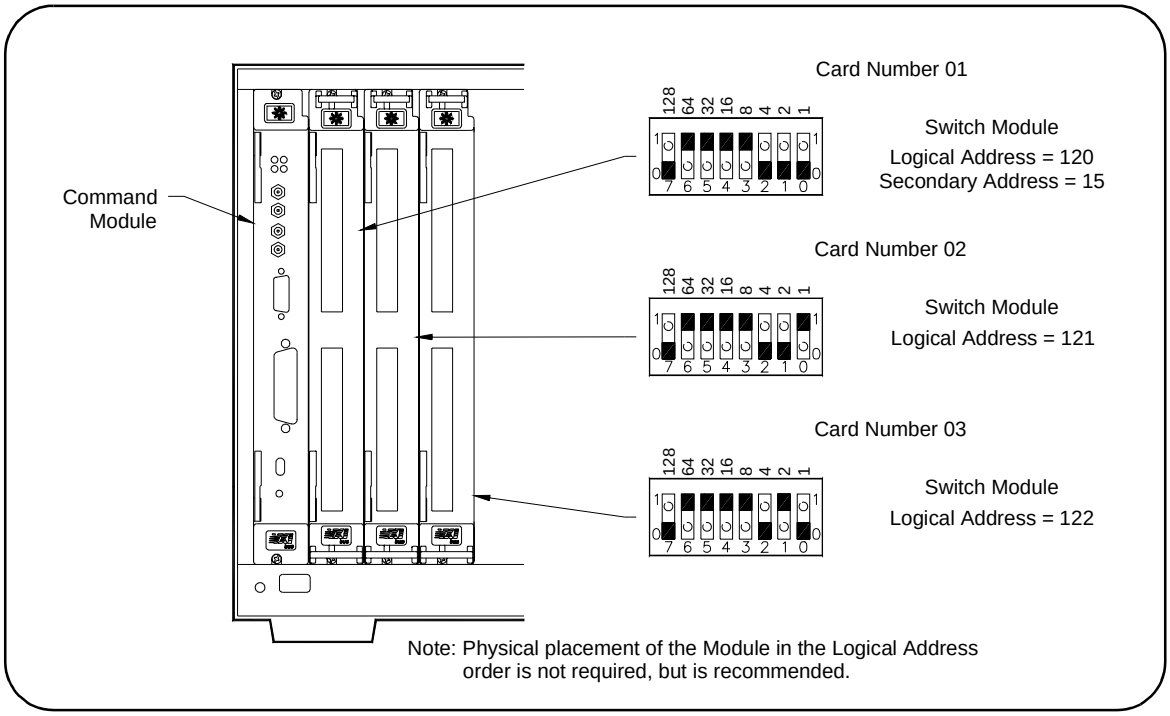


Figure 1-24. Multiple-Module Switchbox Card Numbers

Channel Addresses

Channel addresses (*channel_list*) have the form (@*ccnn*) where *cc* = switch card number (01-99) and *nn* = channel numbers (00-63). You can address single channels (@*ccnn*), multiple channels (@*ccnn*,*ccnn*,...), sequential channels (@*ccnn*:*ccnn*), groups of sequential channels (@*ccnn*:*ccnn*,*ccnn*:*ccnn*) or any combination.

Form C switch channel numbers are 00 through 63. The channels can be addressed using channel numbers or channel ranges. For a single-module switchbox, channel ranges can span across the channels. For multiple-module switchboxes, channel ranges can span across the channels of all modules.

Use commas (,) to form a channel list or use a colon (:) to form a channel range. Only valid channels can be accessed in a channel list or channel range. The channel list or 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. Some example channel lists/ranges follow.

```
CLOS (@100,112)      ! Close channels 00 and 12 on card 01
OPEN (@203,210)      ! Open channels 03 and 10 on card 02
OPEN (@100:163)      ! Open all channels on card 01
SCAN (@100:163)      ! Scan all channels on card 01
SCAN (@100:199)      ! Scan all channels on card 01
```

Start-Up Exercises

This section provides a set of four start-up exercises you can use to quickly get your E1442A 64-Channel Form C Switch operational, including:

- Exercise 1: Check Device Driver (E1406A only)
- Exercise 2: Query Module Identity
- Exercise 3: Perform Open, Close, and Scan Operations
- Exercise 4: Check for System Errors

NOTE *We recommend you do not make user connections to the switch until you have verified correct switch operation. If you have already connected user inputs to the terminal module, you may want to remove the terminal module from the switch module while doing these exercises.*

Exercise 1: Check Device Driver

If you use an E1406 Command Module, you can check the command module for the correct version of the "SWITCH" device driver for the E1442A. Skip this step and go to Exercise 2 if you do not use an E1406 Command Module.

Power-up the mainframe with the command module installed. The command module is the resource manager at logical address 0 and is typically addressed in the mainframe by 70900. Input this BASIC program into your computer.

```
10  DIM A$[256]
20  OUTPUT 70900;"DIAG:DRIV:LIST?"
30  ENTER 70900;A$
40  PRINT A$
50  END
```

RUN the program and look for the device driver "SWITCH,SWITCHBOX,A.08.00,RAM". RAM could be FLASH (flash ROM) depending on where the device driver is loaded. DIAGnostic:DRIVer:LIST? queries the command module at address 70900 for a list of the device drivers loaded in the command module. A typical response should be similar to the following and will depend on the specific drivers that were previously loaded in the command module.

```
SYSTEM,E1406A,A.08.00,ROM;A.04.02,ROM;VOLTMTR,E1326A,
A.06.00,ROM;SWITCH,SWITCHBOX,A.08.00,RAM;COUNTER,
E1332A,A.04.02,ROM;E1333A,A.04.02,ROM;DIG_I/O,E1330A,
A.04.03,ROM;D/A,E1328A,A.04.02,ROM
```

The SWITCH version A.08.00 driver (or later) must appear in this list for the E1442A. If not, you must load a new device driver. To load a new version device driver, you need your device driver version A.08.00 disk and the *Installing SCPI Device Drivers* (part number E1401-90022).

NOTE *For the latest information on instrument drivers, see http://www.agilent.com/find/inst_drivers.*

Exercise 2: Query Module Identity

Turn mainframe power OFF. If you want to set a logical address other than the factory-set address of 120, see "Setting the Logical Address" to set a different logical address for the switch. Install the switch module in the mainframe. See "Installing the Switch in a Mainframe" for steps to install the switch.

NOTE *If you have already connected user inputs to the terminal module, you may want to disconnect the terminal module from the switch module for this exercise. See "Attaching Terminal Modules to the Switch Module" to disconnect the terminal module.*

Turn mainframe power ON and enter the following BASIC program into your computer. For this program, the GPIB Select Code = 7, the primary address = 09, and the logical address = 120. The logical address divided by 8 = the secondary address (120/8 = 15). Thus, the instrument address is 70915.

```
10 DIM A$[256]
20 OUTPUT 70915;"*IDN?"
30 ENTER 70915;A$
40 PRINT A$
50 END
```

RUN the program. The response should be as follows. The device driver revision must be A.08.00 or later.

"HEWLETT PACKARD,SWITCHBOX,0,A.08.00"

Exercise 3: Perform Open, Close, and Scan Operations

This exercise performs close, open and scanning operations and queries the status byte. Now that communication with the module has been established, you can perform some close, open and scan operations and use the "SCAN COMPLETE" bit in the Status Operation Event register (bit 8).

Operation Event Register bit 8 designates scan complete when high. Reading this register clears the register (all bits to zero). This bit is monitored by serial polling (SPOLL) the status byte register (bit 7) in line 70. You may want to look at the STATUS command in Chapter 3 which graphically shows the relationship of these two bits and all status registers relating to this module.

Input this BASIC program into your computer. Do not input the comments preceeded by " ! ".

```
10  DIM A$[256]                                  !Dimension array to hold data entered
20  OUTPUT 70915;"CLOSE (@100, 101, 102:163)"    !Close all channels
30  OUTPUT 70915;"*RST"                          !Open all channels by resetting module
40  OUTPUT 70915;"STAT:OPER:ENAB 256"            !Enable bit 8 of status operation event register
50  OUTPUT 70915;"SCAN (@100:163)"               !Scan all channels
60  OUTPUT 70915;"INIT"                          !Initiate the scan using the default TRIG[:IMM]
70  WHILE NOT BIT (SPOLL(70915),7)               !Serial poll bit 7 of the status byte until it is high
80  PRINT "WAITING FOR SCAN COMPLETE"
```

```
90   END WHILE
100  OUTPUT 70915;"STAT:OPER?"                    !Query the status operation event register
110  ENTER 70915;A$                               !Bit 8 reported high (status byte bit 7 was high)
120  PRINT "STAT:OPER:EVENT BIT 8 = ",A$          !Print response to the STAT:OPER query
130  END
```

RUN the program. You should hear channel relays opening and closing, especially when a large channel list is scanned.

Exercise 4: Check for System Errors

You can add the following lines to the program in Exercise 3 to verify that no system errors were generated. It is always a good idea to check if your program causes the instrument to report any errors during program development (such as command strings that are invalid and cause an error to be sent to the instrument's error queue). You can read the instrument's error queue by inserting the following four program lines (all errors are read until the error queue is `"+0, No errors"`).

```
121  REPEAT
122     OUTPUT 70915;"SYST:ERR?"
123     ENTER 70915; A,A$          A gets the error number, A$ gets the error message
124     PRINT A,A$
125  UNTIL A=0
```

See "Using Interrupts With Error Checking" in Chapter 2 for detecting errors with interrupts. For example, inserting the following (incorrect) program line:

```
51 OUTPUT 70915;"TRIG:SOURC BUS"
```

will cause an error to be sent to the error queue because `TRIG:SOURC BUS` is an incorrect command header (must be `TRIG:SOUR BUS`). The instrument still functions using the default value `TRIG:IMMediate`. To know that an error was reported and your instrument is doing what you intended it to do, you must read the error register with a `SYSTem:ERRor?` command.

You can insert this program segment at different places in your program to see where the error is generated when debugging your program if it cannot be determined from the error message or by examining the program lines. In this case, the error is returned as `-113, "Undefined header"` which means the command header was incorrectly specified. This error is generated by the instrument driver while trying to parse the command (the error `-113` is documented in the command module manual).

Notes:

Chapter 2
E1442A Application Examples

Using This Chapter

This chapter provides application information and examples for using the E1442A 64-Channel Form C Switch Module in a switchbox. The chapter contents are:

General Scanning Information

This section lists general scanning information for the E1442A module, including:

- Switchbox Definition
- How to Scan
- Reset Conditions
- Using Scanning Trigger Sources
- Using the Scan Complete Bit

Switchbox Definition

A switchbox can consist of a single-switch module or multiple-switch modules. It can also include other switch modules that are controlled by the same SWITCH device driver. Figure 2-1 shows a typical switchbox consisting of three cards (modules).

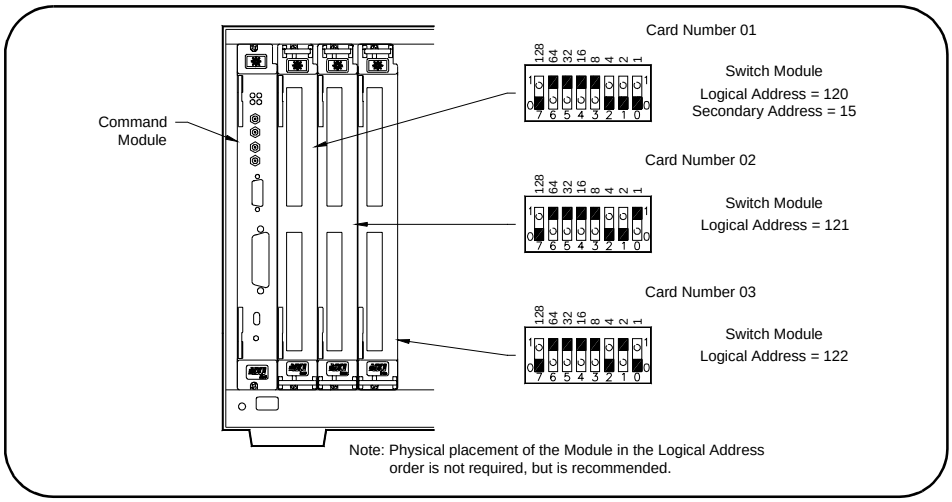


Figure 2-1. Typical Switchbox Configuration

How to Scan

Scanning Form C switch channels consists of closing a set of channels (connecting NO to C) one channel at a time. Single scan, multiple (ARM:COUNt 2 to ARM:COUNt 32767) scans, or continuous INIT:CONT) scanning modes are available. See the command reference in Chapter 3 for more information on these commands. Table 2-1 shows a number of SCPI commands that relate to scanning.

Command	Description
ARM:COUNt	Sets the number of scanning cycles per INIT (optional).
INIT	Begins scanning (required).
INIT:CONTinuous ON	Selects continuous scanning (optional).
OUTPut[:EXTernal] [:STATe] ON	Selects Trig Out port (optional).
OUTPut:STATe	Enables/disables Trig Out signal (optional).
SCAN	Defines channels to be scanned (required).
TRIG	Advances to next channel in scan list (required if using HOLD or BUS trigger sources).
TRIGger:SOURce	Sets the trigger source for scan advance (optional).

Reset Conditions

At power-on or following the reset of the module (*RST command), all 64 channels are open (common connected to the normally closed terminal). In addition, after a *RST command the current scan channel list is invalidated. Table 2-2 lists the parameters and default values following power-on or reset.

Parameter	Default	Description
ARM:COUNt	1	Number of scanning cycles is one.
TRIGger:SOURce	IMM	Will advance scanning cycles automatically.
INITiate:CONTinuous	OFF	Number of scanning cycles is set by ARM:COUNt.
OUTPut:[:STATe]	OFF	Trigger output from EXT, TTL or ECL sources is disabled.
Channel State	All 64 channels are open (channels 00 - 63).	
Channel list from SCAN command (after *RST)	Current channel list is invalidated following a reset of the module with the *RST command.	

Using Scanning Trigger Sources

The TRIG:SOUR command specifies the source to advance the scan. You can use the TRIG command to advance the scan when TRIG:SOUR BUS or TRIG:SOUR HOLD is set. The OUTPut command can be used to enable the E1406A Command Module Trig Out port.

Using the Scan Complete Bit

You can use the Scan Complete bit (bit 8) in the Operation Status Register 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. See the STATe:OPERational[: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 GPIB 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 Common command to assert a GPIB 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 completes. This example uses BASIC as the programming language. The computer interfaces with an E1406 Command Module over GPIB. The GPIB select code is 7, the GPIB primary address is 09, and the GPIB secondary address is 15.

```
10  OUTPUT 70915;"*CLS"                    !Clear all switchbox status structure
20  OUTPUT 70915;"STAT:OPER:ENAB 256"      !Enable Scan Complete Bit to set bit 7 in Status Reg
30  OUTPUT 70915;"*SRE 128"                !Enable bit 7 of Status Register to assert SRQ
40  OUTPUT 70915;"TRIG:SOUR EXT"           !Set to external trigger mode
50  OUTPUT 70915;"SCAN (@100:147)"         !Select channels to be scanned
60  OUTPUT 70915;"INIT"                    !Start scanning cycle
70  WHILE NOT BIT(SPOLL(70915),7)          !Waiting for scan complete
80    PRINT "DO OTHER OPERATION HERE"      !Enter program lines for computer to do other oper
90  END WHILE
100 PRINT "INTERRUPT GENERATED"            !Program goes to this line after interrupt is generated
                                           !by a completed scanning cycle.
```

Saving and Recalling States

This section contains information about saving and recalling a switch module state. The switchbox driver can store up to 10 states.

Saving States

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

- Channel Relay State (channels 00 through 63 open or closed)
- ARM:COUNt
- TRIGger:SOURce
- OUTPut[:STATe]
- INITiate:CONTinuous

Recalling States

The *RCL <*numeric_state*> command recalls a previously saved state. Enter the number (0-9) in the state parameter of the desired saved state. If *SAV was not previously executed using the selected number, the switch module will configure to the reset values (see Table 2-2).

NOTE *Scan lists are not saved when a state is saved. You must re-enter your scan list after recalling a state.*

Detecting Error Conditions

There are two general approaches to error checking: polling and using interrupts. This section describes these approaches and shows an example of each approach.

Example: Error Checking Using Polling

The simplest, but most time consuming, approach to error checking is to ask the instrument whether there are errors at every step of the switching process. This is called "polling" and is illustrated in the following example.

```
10  DIM Err$[256]
20  OUTPUT 70915;"CLOS (@101)"          !Close channel 1 switch
30  OUTPUT 70915;"SYST:ERR?"            !Query for error
40  ENTER 70915;Err$                    !Read response
50  IF VAL (Err$) > 0 THEN              !If an error is found (Err$ not 0)
60  PRINT "Error";Err$                  !Print the error
70  STOP                                !Quit if error encountered
80  END IF
90  ... (PROGRAM CONTINUES)
```

Example: Error Checking Using Interrupts

The second approach to error checking involves the use of interrupts. The following program is a method of checking for errors using interrupts as you program the switch module. The program monitors the switch's Standard Event Status Register for an error condition.

If no errors occur, the switch module functions as programmed. If errors do occur, the switch module interrupts the computer, and the error codes and messages are read from the error queue. This BASIC programming example has a single switch module at address 70915.

```
10
20
30
40  ON INTR 7 CALL Errmsg                  !Call to print out error message
50  ENABLE INTR7:2
60
70
80
90
100 OUTPUT 70915;"*SRE 32"                 !Enables the standard event summary bit SRE
110 OUTPUT 70915;"*ESE 60"                 !Enables all parser generated errors
                                           !See STATus command figure
120
130
140
 .
 .
190 END
200 !
210
220
230
240
250 SUB Errmsg                             ! Define interrupt service routine
260    DIM A$[256]                         ! Declare response string
270    CLEAR 70915                         ! Clear the switch module
280    B = SPOLL(70915)                    ! Fetch status byte
290    REPEAT                              ! Repeat
300       OUTPUT 70915;"SYST:ERR?"         ! Query for error
310       ENTER 70915;Code,A$              ! Read response
320       PRINT Code,A$                    ! Print error
330    UNTIL Code=0                        ! Keep querying for an error until error code = 0
340    OUTPUT 70915;"*CLS"                 ! Clear status registers/error queue
350    STOP
360 SUBEND
```

Scanning with External Instruments

Scanning Form C switch channels has the same effect as executing multiple CLOSe commands. Thus, scanning is useful when the outputs from a number of devices under test (DUTs) are to be measured with an instrument. Three examples using BASIC programming language follow.

Example: Scanning with External Device

This example uses the E1406 Command Module Trig Out port to synchronize the Form C switch channel closures to an external measurement device. See Figure 2-2 for typical user connections.

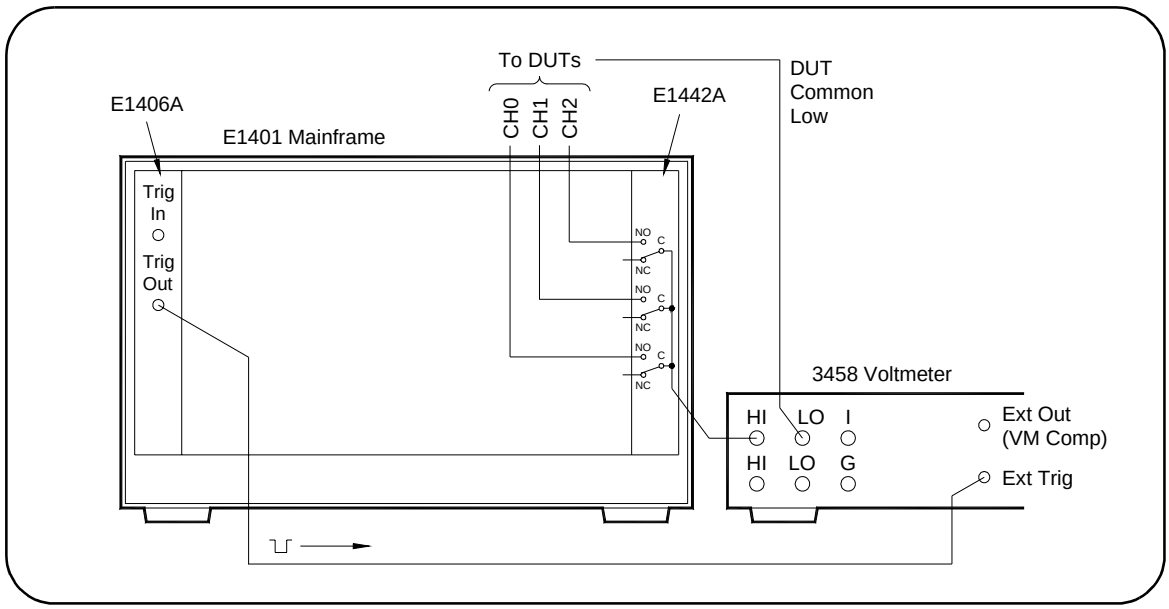


Figure 2-2. Example: Scanning with an External Device

For measurement synchronization, the E1406A Trig Out port is connected to the external instrument (3458 Voltmeter) External Trigger In port. For this example, the mainframe and instrument are connected via GPIB with the mainframe at address 709 and the measurement instrument at address 722.

The Form C switch is at logical address 120 (secondary address 15 and therefore address through the mainframe at address 70915). The measurements are transferred directly to the computer. Appropriate instrument commands must be added to line 10. Also, you may need to add a WAIT statement as line 65 for long measurements. The sequence of operations is:

1. INIT (line 50) closes channel 100.
2. Closure causes trigger output from the Trig Out port.
3. Trigger to Ext Trig In initiates channel 100 measurement.
4. Result is sent to the computer (lines 60-80).
5. TRIGGER (line 90) advances the scan to channel 101.
6. Steps 2-5 are repeated for channels 101-102.

```
10  OUTPUT 722;"TRIG EXT;...."          ! Configure instrument
20  OUTPUT 70915;"OUTP ON"              ! Enable Trig Out port
30  OUTPUT 70915;"TRIG:SOUR BUS"        ! GPIB bus triggering
40  OUTPUT 70915;"SCAN (@100:102)"      ! Scan channels 00-02
50  OUTPUT 70915;"INIT"                 ! Enable scan.
60  FOR I=1 TO 3                        ! Start count loop
70     ENTER 722;A                      ! Enter reading
80     PRINT A
90     TRIGGER 70915                    ! Advance scan
100 NEXT I                             ! Increment count
110 END
```

Example: Scanning Using Trig Out and Trig In Ports

This example uses the E1406A Command Module Trig Out and Trig In ports to synchronize Form C switch channel closures with an external measurement device. See Figure 2-3 for typical user connections.

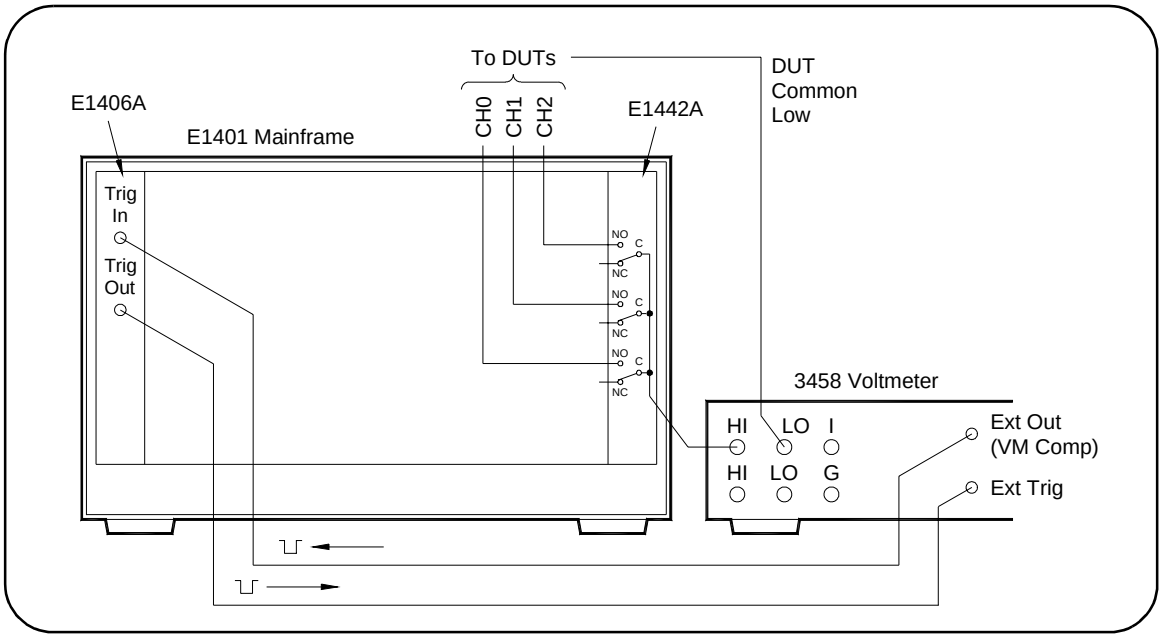


Figure 2-3. Example: Scanning Using Trig Out and Trig In Ports

For this example, the mainframe and measurement instrument are connected via GPIB with a mainframe at address 709 and the measurement instrument at address 722. The Form C switch logical address is 120 (secondary address = 120/8 = 15 and therefore addressed through the mainframe at 70915).

With this example, since synchronization with the computer cannot be ensured, the external instrument must have internal memory capacity to store the readings. Also, you must add the appropriate instrument commands to line 10. The sequence of operation is:

1. INIT (line 50) closes channel 100.
2. Closure causes trigger to be output from Trig Out port.
3. Trigger to Ext Trig In initiates channel 100 measurement.
4. Channel 100 measurement result stored in instrument.
5. Trigger is then output from Measurement Complete port.
6. Trigger to Event In port advances scan to channel 101.
7. Steps 2-6 are automatically repeated for channels 101-102.

```
10  OUTPUT 722;"TRIG EXT; .... "          ! Configure voltmeter
20  OUTPUT 70915;"OUTP ON"                ! Enable Trig Out port
30  OUTPUT 70915;"TRIG:SOUR EXT"          ! Event In triggering
40  OUTPUT 70915;"SCAN (@l00:102)"        ! Scan channels 00-02
50  OUTPUT 70915;"INIT"                   ! Enable scan.
60  FOR Chan = 1 to 3
70      PRINT "Channel", Chan, Result
80  NEXT Chan
90  OUTPUT 70915;"*RST"                   ! Reset module and open last
                                          ! switch closed
100 END
```

Example: Synchronizing the Form C Switch

This example discusses synchronizing the switch to other instruments when making measurements. The following example uses the switch module to switch a signal to be measured by a multimeter. The program verifies that the switching is complete before the multimeter begins a measurement.

The measurement setup consists of a Digital Multimeter with a GPIB select code = 7, primary address = 09 and secondary address = 03 (addressed as 70903) and an E1442A with a GPIB select code = 7, primary address = 09 and secondary address = 15 (addressed as 70915).

```
10  OUTPUT 70915;"CLOS  (@100)"           ! Close channel 100
20  OUTPUT 70915;"*OPC?"                  ! Wait for completion of close ! command
30  ENTER 70915;Opc_value                 ! Read response to *OPC? command.
31  !
32  ! Channel is closed and the measurement  can be made.
33  !
40  OUTPUT 70903;"MEAS:VOLT:DC?"          ! Make VM measurement
50  ENTER 70903;Meas_value                ! Read the measurement
60  PRINT Meas_value                      ! Print the measurement
70  END
```

Chapter 3
E1442A Command Reference

Using This Chapter

This chapter describes Standard Commands for Programmable Instruments (SCPI) and summarizes IEEE 488.2 Common (*) commands applicable to the E1442A 64-Channel Form C Switch Module. This chapter contains the following sections:

Command Types

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

Common Command Format

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

```
*RST    *ESR 32    *STB?
```

SCPI Command Format

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

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

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

Command Separator

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

[ROUTe:]SCAN:MODE?

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

Abbreviated Commands

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

For example, if the command syntax shows MEASure, then MEAS and MEASURE are both acceptable forms. Other forms of MEASure such as MEASU or MEASUR will generate an error. You may use upper- or lowercase letters. Therefore, MEASURE, measure, and MeAsUrE 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 [ROUTe:] subsystem shown below:

```
[ROUTe:]
    CLOSe<channel_list>
    CLOSe?<channel_list>
    OPEN<channel_list>
    OPEN?<channel_list>
    SCAN<channel_list>
        :MODE NONE|VOLT
        :MODE?
```

The root command [ROUTe:] is an implied command (indicated by square brackets [< >]). To close relays in a channel list, you can send either of the following command statements:

[ROUTe:]CLOSe (@100:107, 201, 225) *or* CLOSe (@100:107, 201, 225)

These commands function the same closing channels 00 through 07 on card 1 and channels 01 and 25 on card 2.

Parameters

ParameterTypes. The following table contains explanations and examples of parameter types you might see later in this chapter.

Type	Explanations and Examples
Numeric	Accepts all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation. Examples are 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01. Special cases include MIN, MAX and INF.
Boolean	Represents a single binary condition that is either true or false. (ON, OFF, 1.0).
Discrete	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, EXT, HOLD, or IMM.

Optional Parameters. Parameters shown within square brackets ([ ]) are optional parameters. *(Note that the brackets are not part of the command and are not sent to the instrument.)* If you do not specify a value for an optional parameter, the instrument chooses a default value.

For example, consider the ARM:COUNt? [MIN |MAX] command. If you send the command without specifying a parameter, the present setting is returned. If you send the MIN parameter, the command returns the minimum count available. If you send the MAX parameter, the command returns the maximum count available. Be sure to place a space between the command and the parameter.

Linking Commands

Linking IEEE 488.2 Common Commands with SCPI Commands. Use a semicolon (;) between the commands. For example, *RST;OUTP ON or TRIG:SOUR HOLD;*TRG

Linking Multiple SCPI Commands. Use both a semicolon (;) and a colon (:) between the commands. For example, ARM :COUN 1;: TRIG:SOUR EXT

SCPI Command Reference

This section describes the Standard Commands for Programmable Instruments (SCPI) commands for the E1442A. Commands are listed alphabetically by subsystem and within each subsystem.

There are two methods to send commands to the instrument. The first method is from a controller over the GPIB interface. This method will be referred to as the "GPIB interface" in the command reference. The second method of sending commands is from a terminal connected to the E1406 Command Module (RS-232). Commands sent this way will be referred to as "from the terminal" in the command reference.

ABORt

The ABORt command stops a scan in progress when the trigger sources are either TRIGger:SOURce BUS or TRIGger:SOURce HOLD. See the comments to stop a scan if trigger source is not BUS or HOLD.

Subsystem Syntax

ABORt

Comments

Channel Status After an ABORt: ABORting a scan will leave the last channel that it closed in the closed position.

Effect on Scan Complete Status Bit: ABORting a scan will not set the "scan complete" status bit.

Stopping Scans Enabled from GPIB Interface: When a scan is enabled from the GPIB interface, and the trigger source is not HOLD or BUS, you can clear the interface to stop the scan (in the BASIC programming language, this is done by executing the CLEAR command for your interface (such as CLEAR 7).

When the scan is enabled from the GPIB interface and the trigger source is TRIGger:SOURce BUS or TRIGger:SOURce HOLD, send the ABORt command over the GPIB bus.

NOTE *Clearing the GPIB interface during a scan leaves the last channel the scan closed in the closed position and does not set the "scan complete" status bit.*

Stopping Scans by Using the RS-232 Terminal: You may use a terminal connected to the E1406 Command Module to stop any scan.

If the scan was started from the terminal, and the trigger source is HOLD or BUS, send the ABORt command to halt the scan. If the scan was started from the terminal and some other trigger source is being used, a **Ctrl+C** will send an interface CLEAR to the instrument and abort the scan. Sending **Ctrl+R** also sends an interface CLEAR to the instrument and additionally performs a reset (*RST) on the instrument. (See the Command Reference in the command module's user's manual for details on the terminal interface.)

If the scan was started from the GPIB interface but you want to stop it by using the terminal, first make sure that the correct instrument (SWITCH at desired logical address) is selected by using the terminal soft keys. Then, send a **Ctrl+R**. This will send an interface CLEAR to the GPIB task, but will not place the instrument in the reset state with respect to the GPIB task. These actions will occur regardless of the trigger source setting.

NOTE *Clearing the interface using a **Ctrl+C** from the terminal during a scan leaves the last channel it closed in the closed position and does not set the Scan Complete status bit.*

Related Commands: ARM, INITiate:CONTinuous, [ROUTe:]SCAN, TRIGger

Example

Stopping a Scan with ABORt

```
TRIG:SOUR BUS                 Bus is trigger source
INIT:CONT ON                  Sets continuous scanning
SCAN  (@100:115)              Sets channel list
INIT                          Starts scanning cycle
  .
  .
ABOR                          Aborts scan in progress
```

ARM

The ARM subsystem allows a scan list to be scanned multiple times (1 through 32767) with one INITiate command.

Subsystem Syntax

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

ARM:COUNt

ARM:COUNt <*number*> allows scanning cycles to occur a multiple of times (1 to 32767) with one INITiate command and when INITiate:CONTinuous OFF | 0 is set.

Parameters

Name	Type	Range of Values	Default Value
<*number*>	numeric	1 - 32767 | MIN | MAX	1

Comments

Number of Scans: Use only values between 1 (MIN) to 32767 (MAX) for the number of scanning cycles (default is MIN = 1).

Related Commands: ABORt, INITiate:IMMediate, INITiate:CONTinuous

***RST Condition:** ARM:COUNt 1

Example

Setting Ten Scanning Cycles

```
ARM:COUN 10          Sets 10 scanning cycles
SCAN  (@100:115)     Sets channel list
INIT                 Starts scanning cycle
```

ARM:COUNt?

ARM:COUNt? [MIN | MAX] returns the current number of scanning cycles set by ARM:COUNt. If a value between MIN and MAX is set, that value for ARM:COUNt is returned. The optional parameters MIN and MAX allow you to query the module for the these values instead of looking them up in the command reference. A 1 is returned for the MIN parameter. 32767 is returned for the MAX parameter regardless of the ARM:COUNt value set.

Parameters

Name	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 Scanning Cycles

```
ARM:COUN 55                 Set 10 scanning cycles
ARM :COUN?                  Query number of scanning cycles.
                            Returned value is 55.
```

DISPlay

The DISPlay subsystem monitors the channel state of a selected module (or card) in a switchbox. The DISPlay command subsystem only operates with an RS-232 terminal connected to the E1406 Command Module's RS-232 port. These commands control the display on the terminal, and would in most cases be typed directly from the terminal keyboard.

However, it is possible to send these commands over the GPIB interface and control the terminal's display. In this case, care must be taken that the instrument receiving the DISPlay command is the same one that is currently selected on the terminal. Otherwise, the GPIB command will have no visible effect.

Subsystem Syntax

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

DISPlay:MONitor:CARD

DISPlay:MONitor:CARD <*number*>|AUTO selects the module in a switchbox to be monitored. You must use DISP:MON:STAT ON to actually display the monitored module state to the RS-232 terminal.

Parameters

Name	Type	Range of Values	Default Value
<*number* | *AUTO*>	numeric	1 - 99	AUTO

Comments

Selecting a Specific Module to be Monitored: Send the card number in a switchbox with the DISPlay:MONitor:CARD command.

Selecting the Present Module to be Monitored (AUTO): Use the DISPlay:MONitor AUTO command to select the last module addressed by a switching command (e.g., [ROUTe:]CLOSe).

***RST Condition:** DISPlay:MONitor:CARD AUTO

Example

Select Module #2 in a Switchbox for Monitoring

```
DISP:MON:CARD 2
```

Selects module #2 in a switchbox

DISPlay:MONitor:CARD?

DISPlay:MONitor:CARD? queries the setting of the :MONitor:CARD command and returns the module in a switchbox to be monitored.

DISPlay:MONitor[:STATe]

DISPlay:MONitor[:STATe] <*mode*> turns the monitor mode on or off. When monitor mode is on, the RS-232 terminal display presents an array of values indicating the open/close state of every switch on the module. This display is dynamically updated each time a switch is opened or closed.

Parameters

Name	Type	Range of Values	Default Value
<*mode*>	boolean	0 | 1 | ON | OFF	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 monitor mode off.

Typing in another command on the terminal will cause the DISPlay:MONitor[:STATe] to be automatically set to OFF (0). NOTE: Use of the OFF parameter is useful only if the command is issued across the GPIB interface.

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

Monitor Mode on an E1406 Command Module Display: A typical display for the E1442A 64-Channel Form C Switch with all channels (all relays) closed follows. The "#H" indicates data is in hexadecimal format. Each channel is represented as a bit in the hex value. The channels are grouped into four blocks of 16 channels each.

15-0 #HFFFF 31-16 #HFFFF 47-32 #HFFFF 63-48 #HFFFF

Closing only channel 3 would appear as 15-0: #H0008.

***RST Condition:** DISPlay:MONitor[:STATe] OFF | 0. An *RST also opens all switches on the card. A DISP:MON ON command following a *RST will display the following:

15-0 #H0000 31-16 #H0000 47-32 #H0000 63-48 #H0000

Example **Enabling the Monitor Mode**

```
DISP:MON:CARD 2          Selects module #2 in a switchbox.
DISP:MON 1               Turns the monitor mode on.
```

DISPlay:MONitor[:STATe]?

DISPlay:MONitor[:STATe]? queries the monitor mode. The command returns a `1` if monitor mode is on or a `0` if monitor mode is off.

INITiate

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

Subsystem Syntax

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

INITiate:CONTinuous

INITiate:CONTinuous <*mode*> enables or disables continuous scanning cycles for the switchbox. The setting of this command determines whether or not a subsequent INIT[:IMMediate] command will cause a continuous scan to occur.

Parameters

Name	Type	Range of Values	Default Value
<*mode*>	boolean	0 | 1 | ON | OFF	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 a trigger source selected 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 list and the scan cycle repeats.

Non-Continuous Scanning Operation: Non-continuous 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 a trigger source selected by the TRIGger:SOURce command advances the scan through the channel list. A trigger at the end of the channel list opens the last channel in the list and the scanning cycle stops.

The INITiate:CONTinuous command does not start a scanning cycle (see INIT:IMM).

Stopping Continuous Scans: See the ABORt command.

Related Commands: ABORt, ARM:COUNt, INITiate[:IMMediate], TRIGger, TRIGger:SOURce

***RST Condition:** INITiate:CONTinuous OFF

Example Enabling Continuous Scans

```
INIT:CONT ON          Enables continuous scanning
SCAN  (@100:163)      Sets channel list
INIT                  Starts scanning cycle
```

INITiate:CONTinuous?

INITiate:CONTinuous? queries the scanning state. With continuous scanning enabled, the command returns `1`. With continuous scanning disabled, the command returns `0`.

Example Query Continuous Scanning State

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

INITiate[:IMMediate]

INITiate[:IMMediate] starts the scanning cycle and closes the first channel in the channel list. Successive triggers from the source specified by the TRIGger:SOURce command advance 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. A trigger advances the scan through the channel list. An invalid channel list generates an error (see [ROUTe:]SCAN command).

Stopping Scanning Cycles: See the ABORt command.

Related Commands: ABORt, ARM:COUNt, INITiate:CONTinuous, TRIGger, TRIGger:SOURce

***RST Condition:** None

Example Starting a Single Scan

```
SCAN  (@100:163)      Sets channel list
INIT                  Starts scanning cycle by closing
                      channel 00 and proceeding
```

OUTPut

The OUTPut subsystem enables one trigger line of the E1406 Command Module. It also can disable the active line.

Subsystem Syntax

```
OUTPut
    :ECLTrgn
        [:STATe] <mode>
        [:STATe]?
    [:EXTernal]
        [:STATe] <mode>
        [:STATe]?
    :TTLTrgn
        [:STATe] <mode>
        [:STATe]?
```

OUTPut:ECLTrg*n*[:STATe]

OUTPut:ECLTrg*n*[:STATe] <*mode*> enables (ON or 1) or disables (OFF or 0) the ECL trigger bus pulse on the VXI bus line specified by *n*. There are two ECL trigger lines on the VXI bus allowing valid values for *n* to be 0 and 1.

Parameters

Name	Type	Range of Values	Default Value
n	numeric	0 or 1	N/A
<*mode*>	boolean	0 | 1 | ON | OFF	OFF | 0

Comments

When OUTPut:ECLTrg*n*[:STATe] ON is set, a trigger pulse occurs each time a channel is closed during a scan.

OUTPut:ECLTrg*n*[:STATe]?

OUTPut:ECLTrg*n*[:STATe]? queries the state of the ECL trigger bus line specified by *n*. A 1 is returned if the line is enabled. A 0 is returned if it is disabled. Valid values for *n* are 0 and 1.

OUTput:[EXTernal][:STATe]

OUTPut[:EXTernal][:STATe] *<mode>* enables or disables the Trig Out port on the E1406A Command Module. OUTPut[:EXTernal][:STATe] ON | 1 enables the port and OUTPut[:EXTernal][:STATe] OFF | 0 disables the port.

Parameters

Name	Type	Range of Values	Default Value
<*mode*>	boolean	0 | 1 | ON | OFF	OFF | 0

Comments

Abbreviated Syntax: OUTPut subsystem commands [:EXTernal] and [:STATe] are optional subcommands. The OUTPut command can be abbreviated by executing OUTPut ON or OUTPut OFF.

Enabling Trig Out Port: When enabled, the Trig Out port is pulsed each time a channel is closed during scanning. When disabled, the Trig Out port is not pulsed. The output pulse is a +5 V negative-going pulse.

Trig Out Port Shared by Switchboxes: Once enabled, the Trig Out port may be pulsed by the switchbox each time a channel is closed in a switchbox during scanning. To disable the output for a specific switchbox, send the OUTPut[:EXTernal] [:STATe] OFF or OUTPut[:EXTernal][:STATe] 0 command for that switchbox. The OUTP OFF command must be executed following use of this port to allow other instrument drivers to control the Trig Out port.

Related Commands: [ROUTE:]SCAN, TRIGger:SOURce

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

Example

Enabling Trig Out Port

```
OUTP ON                    Enables Trig Out port for pulse output
```

OUTPut[:EXTernal][:STATe]?

OUTPut:[:EXTernal][STATe]? queries the present state of the Trig Out port on the E1406 Command Module. The command returns `1` if the port is enabled, or `0` if disabled.

Example

Query Trig Out Port State

```
OUTP ON                    Enable Trig Out port for pulse output
OUTP:STAT?                 Query port enable state
```

OUTPut:TTLTrg*n*[:STATe]

OUTPut:TTLTrg*n*[:STATe] <*mode*> enables (ON or 1) or disables (OFF or 0) the TTL trigger bus pulse on the VXI bus line specified by *n*. There are eight TTL trigger lines on the VXI bus (n = 0 through 7).

Parameters

Name	Type	Range of Values	Default Value
n	numeric	0 through 7	N/A
<*mode*>	boolean	0 | 1 | ON | OFF	OFF | 0

Comments

When OUTPut:TTLTrg*n*[:STATe] ON is set, a trigger pulse occurs each time a channel is closed during a scan.

OUTPut:TTLTrg*n*[:STATe]?

OUTPut:TTLTrg*n*[:STATe]? queries the state of the TTL trigger bus line specified by *n*. A `1` is returned if the line is enabled. A `0` is returned if the line is disabled. Valid values for *n* are 0 through 7.

[ROUTe:]

The [ROUTe:] subsystem controls switching and scanning operations for the 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>
        :MODE <mode>
        :MODE?
```

[ROUTe:]CLOSe

[ROUTe:]CLOSe*<channel_list>* activates the Form C switch relay for the channels specified in the *channel_list.* The relay's Common (C) terminal is connected to the Normally Open (NO) terminal. The *channel_list* is in the form (@*ccnn*), (@*ccnn*,*ccnn*), or (@*ccnn:ccnn*) where *cc* = card number (00-99) and *nn* = channel number (00-63).

Parameters

Name	Type	Range of Values	Default Value
<*channel_list*>	numeric	cc00 - cc63	N/A

Comments

Special Case of Using Upper Range 99 in the Channel List: Specifying the last channel as 99 (for example, (@100:199) automatically closes all channels on the card number specified by *cc*.

Closing Channels: To close:

- ROUTe:]CLOSe (@*ccnn*) to close a single channel
- [ROUTe:]CLOSe (@*ccnn*,*ccnn*) to close multiple channels
- [ROUTe:]CLOSe (@*ccnn:ccnn*) to close sequential channels
- [ROUTe:]CLOSe (@*ccnn:ccnn*,*ccnn:ccnn*) to close a group of sequential channels
- any combination of the above

Closure Order: A list of channels will not all close simultaneously. The order channels close when specified from a single command is not guaranteed. Use sequential CLOSe commands if needed.

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

***RST Condition:** All Form C switch channels are open.

Example

Closing Form C Switch Channels

This example closes channel 00 of card number 1 Form C switch module and channel 15 of card number 2 Form C switch module in a single switchbox.

```
CLOS  (@100,215)
```

100 closes channel 00 of Form C switch #1. 215 closes channel 15 of Form C switch #2.

[ROUTe:]CLOSe?

[ROUTe:]CLOSe? <*channel_list*> returns the current state of the channel(s) queried. The *channel_list* is in the form (@*ccnn*). The command returns `1` if the channel is in the NO state (C connected to NO) or returns `0` if the channel is in the NC state (C connected to NC). If a list of channels is queried, a comma-delineated list of 0 or 1 values is returned in the same order of the channel list.

Comments

Query is Software Readback: The [ROUTe:]CLOSe? command returns the current state of the hardware controlling the specified channel. It does not account for a failed switch element or a relay closed by direct register access (see Appendix B).

Example

Query Form C Switch Channel Closure

```
CLOS  (@100,215)
```

100 closes channel 00 of Form C switch #1. 215 closes channel 15 of Form C switch #2.

```
CLOS?  (@215)
```

Query channel 215

[ROUTe:]OPEN

[ROUTe:]OPEN <*channel_list*> de-energizes the relays for the channels specified in the channel_list connecting the Common (C) terminal to the Normally Closed (NC) terminal. The *channel_list* is in the form (@*ccnn*), (@*ccnn*,*ccnn*), or (@*ccnn:ccnn*) where *cc* = card number (00-99) and *nn* = channel number (00-63).

Parameters

Name	Type	Range of Values	Default Value
<*channel_list*>	numeric	cc00 - cc63	N/A

Comments

Using Upper Range 99 in the Channel List: Specifying the last channel as 99 (for example, (@100:199) automatically opens all channels on the card number specified by *cc*.

Opening Channels: To open:

- a single channel, use [ROUTe:]OPEN (*@ccnn)*
- for multiple channels, use [ROUTe:]OPEN (*@ccnn*,*ccnn*)
- sequential channels, use [ROUTe:]OPEN (*@ccnn:ccnn*)
- a group of sequential channels, use [ROUTe:]OPEN (*@ccnn:ccnn*,*ccnn:ccnn*)
- or any combination of the above

Opening Order: A list of channels will not all open simultaneously. The order channels open when specified from a single command is not guaranteed. Use sequential OPEN commands if needed.

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

***RST Condition:** All Form C switch channels are open.

Example

Opening Form C Switch Channels

This example opens channel 00 of a card number 1 Form C switch module and channel 63 of a card number 2 Form C switch module in a single switchbox.

```
OPEN  (@100,263)
```

100 opens channel 00 of Form C switch #1. 263 opens channel 63 of Form C switch #2.

[ROUTe:]OPEN?

[ROUTe:]OPEN? <channel_list> returns the current state of the channel queried. The *channel_list* is in the form (*@ccnn*). The command returns `1` if the channel is in the NC state (C connected to NC) or returns `0` if the channel is in the NO state (C connected to NO). If a list of channels is queried, a comma delineated list of 0 or 1 values is returned in the same order of the channel list.

Comments

Query is Software Readback: The [ROUTe:]OPEN? command returns the current state of the hardware controlling the specified channel. It does not account for a failed switch element.

Example

Query Form C Switch Channel Open State

```
OPEN  (@100,263)
```

100 opens channel 00 of Form C switch #1. 263 opens channel 63 of Form C switch #2.

```
OPEN?  (@263)
```

Query channel 263

[ROUTe:]SCAN

[ROUTe:]SCAN <*channel_list*> defines the channels to be scanned. The *channel_list* is in the form (@*ccnn*), (@*ccnn*,*ccnn*), or (@*ccnn:ccnn*) where *cc* = card number (00-99) and *nn* = channel number (00-63 and 99). See the comments for explanation of using the special case of 99 in the channel list.

Parameters

Name	Type	Range of Values	Default Value
<*channel_list*>	numeric	cc00 - cc63, cc99	N/A

Comments

Special Case of Using Upper Range 99 in the Channel List: Specifying the last channel as 99 (for example, @100:199) automatically scans all channels on the card number specified by *cc*.

Defining the Channel List: When executing [ROUTe:]SCAN, the channel list is checked for valid card and channel numbers. An error is generated for an invalid channel list.

Scanning Operation: With a valid channel list, INITiate[:IMMediate] starts the scanning cycle 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.

Stopping Scan: See the ABORt command.

Related Commands: CLOSe, OPEN, SCAN:MODE, TRIGger, TRIGger:SOURce

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

Example

Scanning Using External Devices

This BASIC language example shows how to scan channels via GPIB using the E1406 Command Module and a 3457A Digital Multimeter. This example uses the command module's Trig Out port to synchronize the switch module in a switchbox to the multimeter.

The trigger pulse from the Trig Out port triggers the multimeter for a measurement. See Chapter 2 for typical user connections to the Form C switch module. The addresses used are 70900 for the E1406 Command Module, 722 for the 3457A Multimeter, and 70915 for the switchbox.

```
10 OUTPUT 722;"TRIG EXT;DCV"          Sets multimeter to external trigger
                                      and to measure dc volts.
20 OUTPUT 70915;"OUTP ON"             Enables Trig Out port on
                                      command module.
30 OUTPUT 70915;"TRIG:SOUR BUS"       Sets switchbox to receive bus
                                      triggers
40 OUTPUT 70915;"SCAN:MODE VOLT"      Sets switchbox to measure
                                      voltage during scanning
50 OUTPUT 70915;"SCAN (@100:163)"     Selects the channel list
60 OUTPUT 70915;"INIT"                Starts scanning cycle
70 FOR I=1 TO 64                      Starts count loop
80      ENTER 722;A                   Enters voltmeter reading into
                                      variable A
90       PRINT A                      Prints reading in variable A
100     TRIGGER 70915                 Triggers the switchbox to advance
                                      the channel list
110 NEXT I                            Increments count
120 END
```

[ROUTe:]SCAN:MODE

[ROUTe:]SCAN:MODE <*mode*> sets the Form C switch channels defined by the [ROUTe:]SCAN <*channel_list*> command for "no measurements".

The SWITCH device driver for the E1442A also supports Form C switches which use this command to close appropriate tree relays for a specific kind of measurement (such as 2-wire and 4-wire ohms that require different tree relay closures).

For compatibility in use with the switchbox device driver, the E1442A accepts the SCAN:MODE command but the command has no effect on Form C operation. It is important to note that the command erases the current SCAN list when executed.

NOTE *This command erases the current SCAN channel list. SCAN:MODE must be followed by a [ROUTe:]SCAN command to re-establish a scan channel list.*

Parameters

Name	Type	Range of Values	Default Value
<*mode*>	discrete	NONE | VOLT	NONE

Comments

Order of Command Execution: If used, [ROUTe:]SCAN:MODE must be executed before [ROUTe:]SCAN <*channel_list*> because SCAN:MODE erases the current SCAN list. The SCAN:MODE command is not needed for Form C Switch operation.

Related Commands: SCAN

***RST Condition:** [ROUTe:]SCAN:MODE NONE

[ROUTe:]SCAN:MODE?

[ROUTe:]SCAN:MODE? returns the current state of the scan mode. The command returns `NONE` or `VOLT` to indicate which mode the scan is set.

STATus

The STATus subsystem reports the bit values of the Operation Status Register (in the command module). 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 <number>
        :ENABle?
        :EVENt]?
    :PRESet
```

Comments

The STATus system contains four software registers that reside in a SCPI driver, not in the hardware (see Figure 3-1) Two registers 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 (RSQ), Standard Event summary bit (ESB), Message Available bit (MAV) and Questionable Data bit (QUE) in the StatusByte 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 STATus:OPERation: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 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, bit 8 will never appear set when STAT:OPER:COND? is queried. However, bit 8 is set with the STAT:OPER:EVENt? query command.

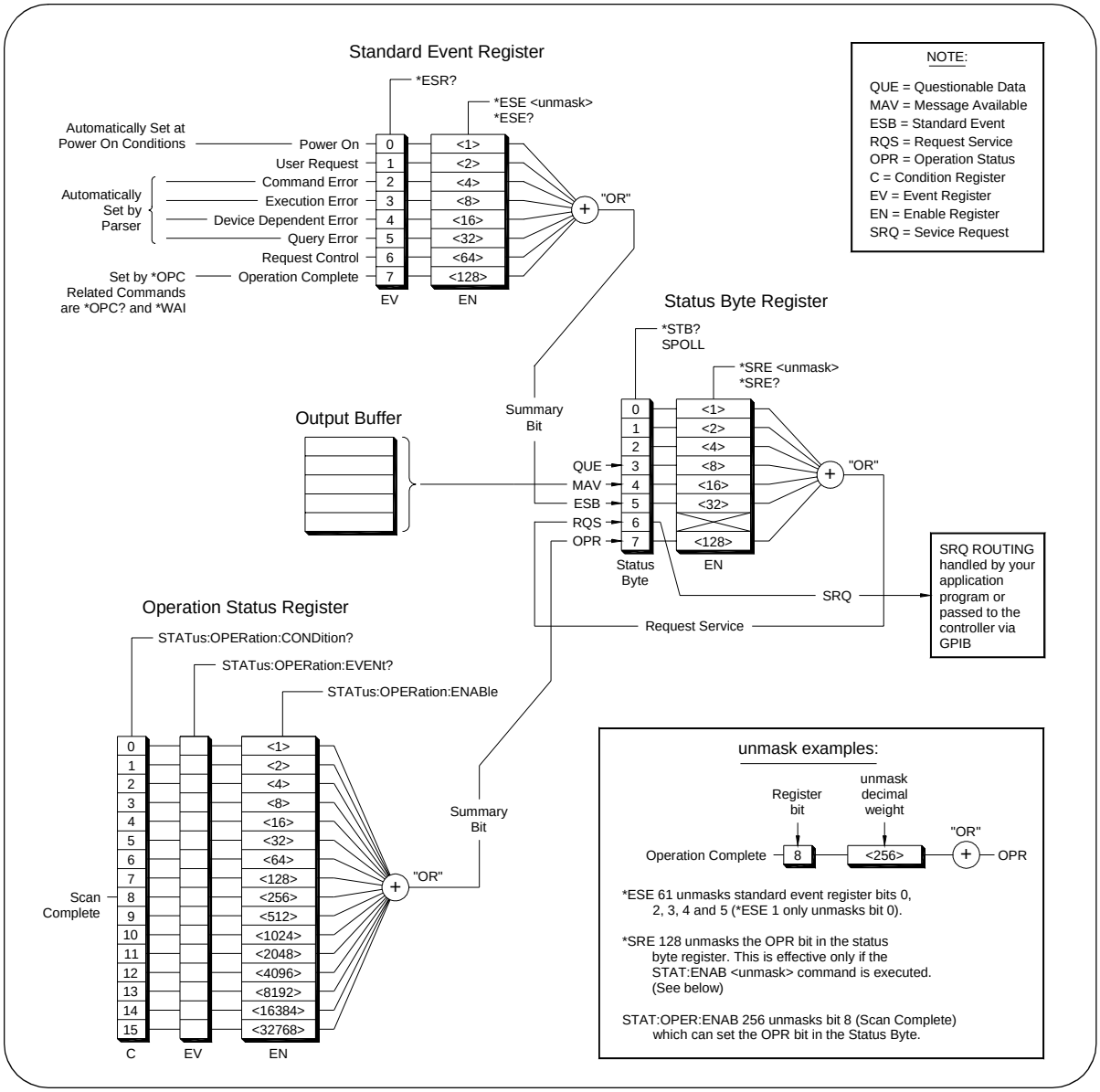


Figure 3-1. E1442A Form C Switch Module Status System

STATus:OPER: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 STAT:OPER:EVENt?).

STATus:OPERation:ENABle

STATus:OPERation:ENABle *<number>* 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 Form C switch modules, when bit 8 in the Operation Status Register is set to 1 and that bit is enabled by the OPER:ENABle command, bit 7 in the Status Register is set to 1.

Parameters

Name	Type	Range of Values	Default Value
<*number*>	numeric	0 through 65535	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

Enable the Status Register

```
STAT:OPER:ENAB 256
```

Enables bit 8 of the Operation Status Register to be reported to bit OPR in the Status Register

STATus:OPERation:ENABle?

STATus:OPERation:ENABle? returns which bits in the Event Register (Operation Status Group) are unmasked.

Comments

Output Format: Returns a decimal weighted value from 0 to 65,535 indicating which bits are set to true. The value returned is the value set by STAT:OPER:ENAB <*number*> 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

STATus: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 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 STATus:OPERation: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?	*Returns the bit values of the Standard Operation Status Register.*
read the register value	*+256 shows bit 8 is set to 1.* *+0 shows bit 8 is set to 0.*

STATus: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, and returns the types and descriptions of modules (cards) in a switchbox.

Subsystem Syntax

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

SYSTem:CDEScription?

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

Parameters

Name	Type	Range of Values	Default Value
<*number*>	numeric	1 through 99	N/A

Comments

Form C Switch Module Description:
For the E1442A, SYSTem:CDEScription?<*number*> returns:

64 Channel General Purpose Switch

Example

Reading the Description of a Card #1 Module

```
SYST:CDES? 1                    Determine the description
```

SYSTem:CPON

SYSTem:CPON *<number>* | ALL opens all channels of a selected or all modules (cards) in a switchbox to their power-on state.

Parameters

Name	Type	Range of Values	Default Value
<*number*>	numeric	1 through 99	N/A

Comments

Differences Between *RST and CPON: SYSTem:CPON only opens all channels of a selected module or all modules in a switchbox. *RST opens all channels of all modules in a switchbox and also sets all other settings to their power-on states.

Example

Set All Channels on Module #1 to Power-on State

```
SYST:CPON 1          Sets module #1 channels to power-on state (open)
```

SYSTem:CTYPe?

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

Parameters

Name	Type	Range of Values	Default Value
<*number*>	numeric	1 through 99	N/A

Comments

Form C Switch Module Model Number: For the E1442A, SYSTem:CTYPe? <*number*> returns:

HEWLETT-PACKARD,E1442A,0,A.08.00

where the four fields of the response are, 1) manufacturer, 2) model number, 3) serial number (always 0), and 4) SWITCH firmware revision.

Example

Reading the Model Number of a Card #1 Module

```
SYST:CTYP? 1         Determine the model number
```

SYSTem: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 the 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. Each error message can be up to 255 characters long but typically is much shorter.

Clearing the Error Queue: An error number/message is removed from the queue each time the SYSTem:ERRor? query command is sent. The errors are cleared first-in, first-out.

When the queue is empty, each following SYSTem:ERRor? query 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 subsystem commands controls the triggering operation of the Form C switch modules in a switchbox.

Subsystem Syntax

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

TRIGger[:IMMediate]

TRIGger[:IMMediate] causes a trigger to occur when the defined trigger source is TRIGger:SOURce HOLD or TRIGger:SOURce BUS. This can be used to trigger a suspended scan operation.

Comments

Executing the TRIGger[:IMMediate] Command: A channel list must be defined in the [ROUTe:]SCAN <*channel_list*> command and an INITiate:IMMediate command must be executed before TRIGger:IMMediate can trigger the switchbox.

HOLD or BUS Source Remains: If selected, theTRIGger:SOURce HOLD or TRIGger:SOURce BUS commands remain in effect, after triggering the switchbox with the TRIGger[:IMMediate] command.

Related Commands: INITiate, [ROUTe:]SCAN, TRIGger:SOURce

Example

Advancing Scan Using the TRIGger Command

```
TRIG:SOUR HOLD          Sets trigger source to hold
SCAN  (@100:163)        Sets channel list
INIT                    Starts scanning cycle
loop statement          Starts count loop
TRIG                    Advances channel list to next
                        channel
increment loop          Increments count loop
```

TRIGger:SOURce

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

Parameters

Name	Type	Range of Values
BUS	discrete	*TRG command
EXTernal	discrete	Trig in port
HOLD	discrete	Hold triggering
ECLTrg*n*	numeric	*n* = 0 or 1
TTLTrg*n*	numeric	*n* = 0 thru 7
IMMediate	discrete	Immediate triggering

Comments

Enabling the Trigger Source: The TRIGger:SOURce command only selects the trigger source. The INIT[:IMMediate] command enables the trigger source. The trigger source must be selected using the TRIGger:SOURce command before executing the INIT command.

One Trigger Input Selected at a Time: Only one input (ECLTrg 0 or 1; TTLTrg 0, 1, 2, 3, 4, 5, 6 or 7; or EXTernal) can be selected at one time. Enabling a different trigger source will automatically disable the active input. For example, if TTLTrg1 is the active input, and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active input.

Using the TRIG command: You can use the TRIGger[:IMMediate] command 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 E1406 Command Module Trig In port.

Using TTL or ECL Trigger Bus Inputs: These triggers are from the VXI backplane trigger lines ECL[0,1] and TTL[0-7]. These may be used to trigger the SWITCH driver from other VXI instruments.

Using EXTernal | TTLTrg*n* | ECLTrg*n* Trigger Inputs: After using TRIGger:SOURce EXT | TTLT*n*| ECLT*n*, the selected trigger source remains assigned to the SWITCH driver until it is relinquished through use of the TRIG:SOUR BUS|HOLD command. While the trigger is in use by the SWITCH driver, no other drivers operating on the E1406 Command Module will have access to that particular trigger source.

Likewise, other drivers may consume trigger resources which may deny access to a particular trigger by the SWITCH driver. You should always release custody of trigger sources after completion of an activity by setting the trigger source to BUS or HOLD (TRIG:SOUR BUS | HOLD).

Using Bus Triggers: To trigger the switchbox with TRIGger:SOURce BUS selected, use the IEEE 488.2 Common command *TRG or the GPIB Group Execute Trigger (GET) command.

Trig Out Port Shared by Switchboxes: See the OUTPut command.

Related Commands: [ROUTe:]SCAN, TRIGger, ABORt

***RST Condition:** TRIGger:SOURce IMMediate

Example Scanning Using External Triggers

In the following example, the trigger input is applied to the E1406 Command Module Trig In port.

```
TRIG:SOUR EXT          Sets trigger source to external
SCAN  (@100:163)       Sets channel list
INIT                   Starts scanning cycle
(trigger externally)   Advances channel list to next channel
```

Example Scanning Using Bus Triggers

```
TRIG:SOUR BUS          Sets trigger source to bus
SCAN  (@100:163)       Sets channel list
INIT                   Starts scanning cycle
*TRG                   Advances channel list to next channel
```

TRIGger:SOURce?

TRIGger:SOURce? returns the current trigger source for the switchbox. Command returns either BUS, EXT, HOLD, TTLT0-7, ECLT0-1 or IMM for trigger sources BUS, EXTernal, HOLD, TTL Trigger, ECL Trigger, ECL Trigger or IMMediate, respectively.

Example Query Trigger Source

```
TRIG:SOUR EXT          Sets trigger source to external
TRIG:SOUR?             Queries trigger source; returns EXT.
```

SCPI Commands Quick Reference

The following table summarizes the SCPI Commands for the E1442A 64-Channel Form C Switch Module used in a switchbox. .

Command		Description
ABORt		Aborts a scan in progress
ARM	:COUNt <*number*> MIN |MAX :COUNt? [MIN|MAX]	Multiple scans per INIT command Queries number of scans
DISPlay	:MONitor:CARD <*number*> |AUTO :MONitor:CARD? :MONitor[:State] ON|OFF|1|0 :MONitor[:State]?	Selects module to be monitored Queries the card number Selects monitor mode Queries the monitor mode
INITiate	:CONTinuous ON | OFF :CONTinuous? [:IMMediate]	Enables/disables continuous scanning Queries continuous scan state Starts a scanning cycle
OUTPut	:ECLTrg*n*[:STATe] ON|OFF|1|0 :ECLTrg*n*[:STATe]? [:EXTernal][:STATe] ON|OFF|1|0 [:EXTernal][:STATe]? :TTLTrg*n*[:STATe] ON|OFF|1|0 :TTLTrg*n*[:STATe]?	Enables/disables the specified ECL trigger line Queries the specified ECL trigger line Enables/disables the Trig Out port on the E1406 Queries the external state Enables/disables the specified TTL trigger line Queries the specified TTL trigger line
[ROUTe:]	CLOSe <*channel _list*> CLOSe? <*channel _list*> OPEN <*channel_list*> OPEN? <*channel _list*> SCAN <*channel_list*> SCAN:MODE NONE|VOLT SCAN:MODE?	Closes channel(s) Queries channel(s) closed Opens channel(s) Queries channel(s) opened Defines channels for scanning Sets scan mode (has no effect on Form C operation) Queries the scan mode
STATus	:OPERation:CONDition? :OPERation:ENABle :OPERation:ENABle? :OPERation[:EVENt]? :PRESet	Returns contents of the Operation Condition Register Enables events in the Operation Event Register to be reported Returns the mask value set by the :ENABle command Returns the contents of the Operation Event Register Enables Register bits to 0
SYSTem	:CDEScription? <*number*> :CTYPe? <*number*> :CPON <*number*> |ALL :ERRor?	Returns description of module in a switchbox Returns the module type Opens all channels on specified module(s) Returns error number/message in a switchbox Error Queue
TRIGger	[:IMMediate] :SOURce BUS :SOURce EXTernal :SOURce HOLD :SOURce IMMediate :SOURce ECLTrg*n* :SOURce TTLTrg*n* :SOURce?	Causes a trigger to occur Trigger source is *TRG Trigger source is Trig In (on the E1406) Holds off triggering Trigger source is the internal triggers Trigger is the VXIbus ECL trigger bus line *n* Trigger is the VXIbus TTL trigger bus line *n* Queries scan trigger source

IEEE 488.2 Common Commands Reference

The following table lists the IEEE 488.2 Common (*) commands accepted by the E1442A 64-channel Form C Switch Module. The operation of some of these commands is described in Chapter 2 of this manual. For more information on Common commands, refer to the user's manual for your mainframe or to the ANSI/IEEE Standard 488.2-1987. The common commands *RCL, *SAV and *TST? do specific actions with the E1442A, as listed in the following table.

Command	Command Description
*CLS	Clears all status registers (see STATus:OPERation[:EVENt]?) and clears the error queue.
*ESE<*unmask*>	Enables Standard Event.
*ESE?	Enables Standard Event Query.
*ESR?	Standard Event Register Query.
*IDN?	Instrument ID Query; returns identification string of the module.
*OPC	Operation Complete.
*OPC?	Operation Complete Query.
*RCL<*n*>	Recalls the instrument state saved by *SAV. You must reconfigure the scan list.
*RST	Resets the module. Opens all channels and invalidates current channel list for scanning. Sets ARM:COUN 1, TRIG:SOUR IMM, and INIT:CONT OFF.
*SAV<*n*>	Stores the instrument state but does not save the scan list.
*SRE<*unmask*>	Service request enable, enables status register bits.
*SRE?	Service request enable query.
*STB?	Read status byte query.
*TRG	Triggers the module to advance the scan when scan is enabled and trigger source is TRIGger:SOURce BUS.
*TST?	Self-test. Executes an internal self-test and returns only the first error encountered. Does not return multiple errors. The following is a list of responses you can obtain where "*cc*" is the card number with the leading zero deleted. +0 if self test passes. +cc01 for firmware error. +cc02 for bus error (problem communicating with the module). +cc03 for incorrect ID information read back from the module's ID register. +cc10 if an interrupt was expected but not received. +cc11 if the busy bit was not held for a sufficient amount of time.
*WAI	Wait to Complete.

Notes:

Appendix A
Specifications

Maximum Input Voltage:

High to Low	Any Terminal to Chassis
150VDC	150VDC
150VAC RMS	150VAC RMS
210VAC Peak	210VAC Peak

(Maximum with internal jumpers installed or use of Option 010 terminal module):

High to Low	Any Terminal to Chassis
60VDC	60VDC
30VAC RMS	30VAC RMS
42VAC Peak	42VAC RMS

Maximum Current (per switch):
1A DC or 1A AC RMS

Maximum Power:
Per Switch: 40W DC, 40VA AC
Per Module: 320W DC, 320VA AC

Thermal Offset: <70 μV per channel

Closed Channel Resistance:
>1.5Ω typical
>13.5Ω at end of relay life

Insulation Resistance:
(between any two points):
$>10^7\Omega$ at 40°C, 65% RH
$>10^8\Omega$ at 25°C, 40% RH

Bandwidth: -3dB at 10 MHz

Crosstalk, Channel to Channel:
>100 kHz: >-70dB
>10 MHz: >-30dB

Capacitance:
Common to NO or NC: >40pF
Channel to Channel: >30pF

Relay Life (typical):
No load: $>10^6$ operations
Max. load: $>10^5$ operations

Power Up/Down States: All Open

Typical Time to Open/Close a Channel: 13 msec

Module Size/Device Type: C, register-based

Installation Category: IC 1

Connectors Used: P1 and P2

Number of Slots: 1

VXIbus Interface Capability: Interrupter, D16

Interrupt Level: 1-7, selectable

Power Requirements:

Peak Module Current			Dynamic Module Current		
Voltage:	+5V	+12V	Voltage:	+5V	+12V
IPM:	0.10 A	0.24A	IDM:	0.11A	0.01A

Watts/Slot: 1.0

Maximum Transient Voltage: 1300V

Operating Temperature: 0° to 55°C

Storage Temperature: -40° to 75°C

Operating Humidity: 40°C and 95% RH

Operating Location: Intended for indoor use only.

IEC Pollution Degree 2 [1] Normally, only non-conductive pollution occurs. Occasionally, however, a temporary conductivity caused by condensation must be expected.

[1] **Pollution**: Any addition of foreign matter, solid, liquid or gaseous (ionized gases), that may produce a reduction of dielectric strength or surface resistivity. **Pollution Degree**: For the purpose of evaluating clearances (the shortest distance in air between two conductive parts), Pollution Degree 1 and Pollution Degree 2 are recognized for use in the micro-environment.

Pollution Degree 1: No pollution or only dry, non-conductive pollution occurs. The pollution has no influence.
Pollution Degree 2: Normally only non-conductive pollution occurs. Occasionally, however, a temporary conductivity caused by condensation must be expected.

Clearance: The shortest distance in air between two conductive parts.

Appendix B
Register-Based Programming

About This Appendix

This appendix contains the information you can use for register-based programming of the E1442A. The contents include:

Register Programming vs. SCPI Programming

The E1442A 64-Channel Form C Switch Module is a register-based module that does not support the VXIbus word serial protocol. When a SCPI command is sent to the Form C switch, the E1406 Command Module parses the command and programs the switch at the register level.

NOTE *If SCPI is used to control this module, register programming is not recommended. The SCPI driver maintains an image of the card state. The driver will be unaware of changes to the card state if you alter the card state by using register writes.*

Register-based programming is a series of **reads** and **writes** directly to the Form C switch registers. This increases throughput speed since it eliminates command parsing and allows the use of an embedded controller. Also, if slot 0, the resource manager, and the computer GPIB interface are provided by other devices, a C-size system can be downsized by removing the command module.

Addressing the Registers

Register addresses for register-based devices are located in the upper 25% of VXI A16 address space. Every VXI device (up to 256 devices) is allocated a 32-word (64-byte) block of addresses. With seven registers, the E1442A Form C Switch Module uses seven of the 64 addresses allocated.

The Base Address

When reading or writing to a switch register, a hexadecimal or decimal register address is specified. This address consists of a base address plus a register offset. The base address used in register-based programming depends on whether the A16 address space is outside or inside the E1406 Command Module.

A16 Address Space Outside the Command Module

When the E1406 Command Module is not part of your VXIbus system (Figure B-1), the switch's base address is computed as:

$C000_{16} + (LADDR * 64)_{16}$ *or* 49,152 + (LADDR * 64)

where $C000_{16}$ (49,152) is the starting location of the register addresses, LADDR is the switch's logical address, and 64 is the number of address bytes per VXI device. For example, the switch's factory-set logical address is 120 (78_{16}). If this address is not changed, the switch will have a base address of:

$C000_{16} + (120 * 64)_{16} = C000_{16} + 1E00_{16} =$ **$DE00_{16}$**

or (decimal)

49,152 + (120 * 64) = 49,152 + 7680 = **56,832**

A16 Address Space Inside the Command Module or Mainframe

When the A16 address space is inside the E1406 Command Module (Figure B-2), the switch's base address is computed as:

$1FC000_{16} + (LADDR * 64)_{16}$ or 2,080,768 + (LADDR * 64)

where $1FC000_{16}$ (2,080,768) is the starting location of the VXI A16 addresses, LADDR is the switch's logical address, and 64 is the number of address bytes per register-based device. Again, the switch's factory-set logical address is 120. If this address is not changed, the switch module will have a base address of:

$1FC000_{16} + (120 * 64)_{16} = 1FC000_{16} + 1E00_{16} =$ **$1FDE00_{16}$**

or

2,080,768 + (120 * 64) = 2,080,768 + 7680 = **2,088,448**

Figure B-1 shows the register address location within A16 as it might be mapped by an embedded controller. Figure B-2 shows the location of A16 address space in the E1406 Command Modules.

Register Offset

The register offset is the register's location in the block of 64 address bytes. For example, the switch's Status/Control Register has an offset of 04_{16}. When you write a command to this register, the offset is added to the base address to form the register address:

$DE00_{16} + 04_{16} = \mathbf{DE04_{16}}$
$1FDE00_{16} + 04_{16} = \mathbf{1FDE04_{16}}$

or

56,832 + 4 = **56,836**
2,088,448 + 4 = **2,088,452**

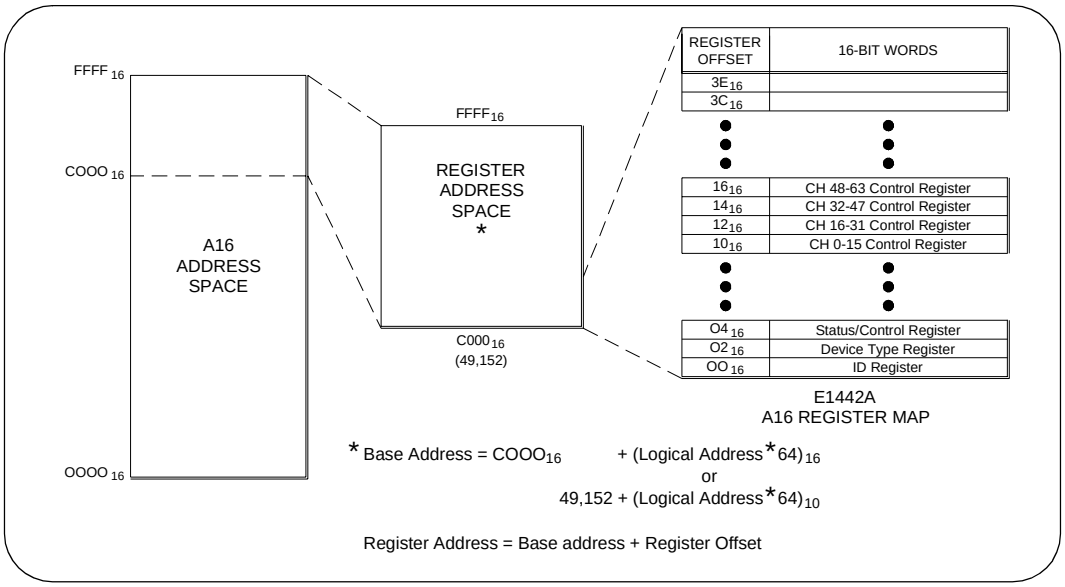


Figure B-1. Registers within A16 Address Space

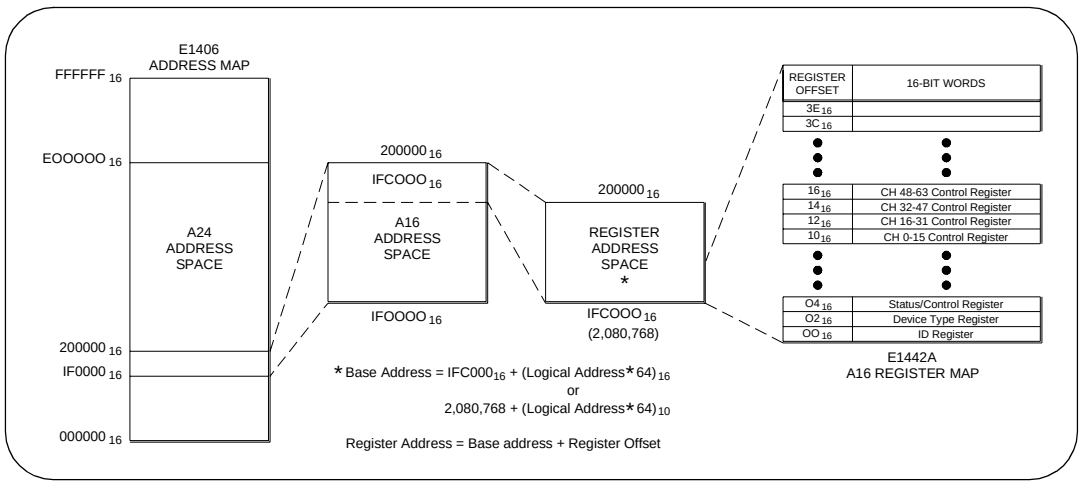


Figure B-2. Registers within the E1406 A16 Address Space

Register-Based Programming the E1442A

The E1442A Form C Switch Module is a register-based slave device. There are 64 independent switches on the card which are controlled using the Switch Control Registers. There are four register types on this module:

- Identifies Hewlett-Packard as the manufacturer and the card is an A16 register-based device.
- Device Type Register - Identifies card as an E1442A.
- Status/Control Register - When read from, it is used to return device-specific status information. When written to, it is used to set control bits.
- Switch Enable Registers - These four registers control the state of the Form C switches on the module (e.g., close or open the switch).

Reading or Writing to E1442A Registers

To read or write to specific registers you must address a particular register within a module. The registers within a module are located using a fixed offset. The module address is based on the module's logical address. There are two basic ways of accessing registers.

One method uses the logical address directly to access a particular card using VXI:READ and VXI:WRITE commands through a command module. The other method can be used with an embedded controller that locates A16 data space within its memory map. The memory mapping allows registers to be directly read or written with moves to/from memory. The factory setting of the logical address switch is 120 (78_{16}). This value is used in the following examples.

Register Access with Logical Address (Command Module)

When using the E1406 Command Module to access registers via VXI:READ and VXI:WRITE commands, the logical address is used to determine which VXI module is being accessed. See the E1406 Command Module documentation for use of the VXI:READ and VXI:WRITE commands and other related commands.

The following commands are sent to the E1406 Command Module via the GPIB. The example below shows a portion of a BASIC program. The controller could either be external or embedded in the VXI mainframe. This example shows the Status/Control Register being accessed.

```
OUTPUT 70900; "VXI:WRITE 120, 4, #HFFFF"        Writes FFFF hex to Control Register

OUTPUT 70900;"VXI:READ 120,4"                   Reads from Status Register
ENTER 70900; Status
```

Register Access with Memory Mapping (Embedded Controller)

When using an embedded controller, VXI A16 address space is usually mapped to some block of memory within the controller's addressable memory space. See the manual for the specific embedded controller you are using to determine where VXI Al6 is mapped. There may be other methods of accessing the VXI backplane. The following method shows which Al6 addresses are calculated for a module.

For example, for a C-size mainframe with an E1406 Command Module, VXI A16 address space starts at $lF0000_{16}$. In the E1406 Command Module, the A16 space is divided so modules are addressed only at locations beginning with $C000_{16}$ within A16. Each module is allocated 64 register addresses (40_{16}). The module base address is related to the logical address set by the logical address switch on the module:

$$(\text{base address})_{16} = (\text{logical address})_{16} * 40_{16} + C000_{16}$$

For the Form C switch, the factory-set logical address is 120 (78_{16}), so to address the Status/Control Register of a Form C switch using the E1406 Command Module:

$$\text{base address} = (78_{16})*(40_{16}) + C000_{16} = DE00_{16}$$

$$\text{register address} = [\text{A16 location}]_{16} + [\text{base address}]_{16} + [\text{register offset}]_{16}$$
$$\text{register address} = 1F0000_{16} + DE00_{16} + 04_{16} = 1FDE04_{16}$$

Reading the E1442A Registers

You can read the following Form C switch registers:

- ID Register (base + 00_{16})
- Device Type Register (base + 02_{16})
- Status/Control Register (base + 04_{16})
- Switch Enable Register for channels 0 - 15 (base + 10_{16})
- Switch Enable Register for channels 16 - 31 (base + 12_{16})
- Switch Enable Register for channels 32 - 47 (base + 14_{16})
- Switch Enable Register for channels 48 - 63 (base + 16_{16})

ID Register

For the Form C switch, a read of the ID Register (base address + 00_{16}) returns $FFFF_{16}$ since the switches are manufactured by Hewlett-Packard and are A16 only, register-based devices. The Device Type Register (base + 02_{16}) returns 0228_{16}.

Device Type Register

For the Form C switch, a read of the Device Type Register (base address + 02_{16}) returns 0228_{16}. This indicates it is a model E1442A.

Status/Control Register

Each relay requires about 13 msec execution time (close to open or open to close) during which time the switch is "busy". A read of the Status/Control Register (base + 04_{16}) returns a 1 in bit 7 when the module is not busy or returns a 0 in bit 7 when the module is busy.

An interrupt is generated after any of the Switch Enable Registers are written. Bit 6 of the Status Register is used to enable/disable interrupts from the card.

If bit 6 is returned as a 0, interrupts are enabled. If bit 6 is returned as a 1, interrupts are disabled.

Bit 14 is the MODID bit. When a 0 is returned in bit 14, the module has been selected with a high state on the P2 MODID line (this occurs during turn-on). If a 1 is returned, the module has not been selected.

Switch Enable Register

A read of any of the Switch Enable Registers always returns $FFFF_{16}$, regardless of the channel states.

Writing to E1442A Registers

You can write to the following Form C switch registers:

- Status/Control Register (base + 04_{16})
- Switch Enable Register for channels 0 - 15 (base + 10_{16})
- Switch Enable Register for channels 16 - 31 (base + 12_{16})
- Switch Enable Register for channels 32 - 47 (base + 14_{16})
- Switch Enable Register for channels 48 - 63 (base + 16_{16})

Status/Control Register

Writing a 1 to bit 0 of the Status/Control Register (base + 04_{16}) to reset the switch module (all channels open). Resetting the module enables interrupts.

NOTE *It is necessary to write a 0 to bit 0 after the reset has been performed before any other commands can be programmed and executed.*

To disable the interrupt generated when channels are opened/closed, write a 1 to bit 6 of the Status/Control Register.

NOTE *Typically, interrupts are disabled when doing register-level access to a module. Refer to the operating manual of the command module or the embedded controller being used to handle interrupts. Interrupts are re-enabled after a reset.*

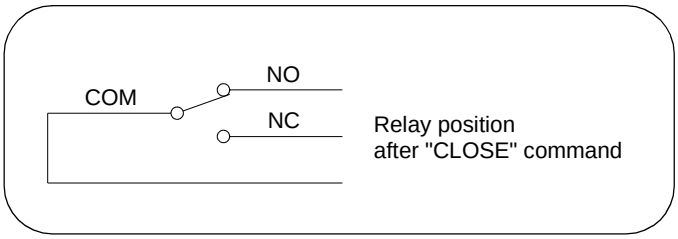
Bit 12 provides status on fuse F4. This is a user-installed component required to provide the +5V pullup voltage to the module's internal bus for the NC and NO contacts. A 0 indicates the fuse is not installed (or the fuse is blown if installed). A 1 indicates you previously installed the fuse and it is good.

Switch Enable Registers

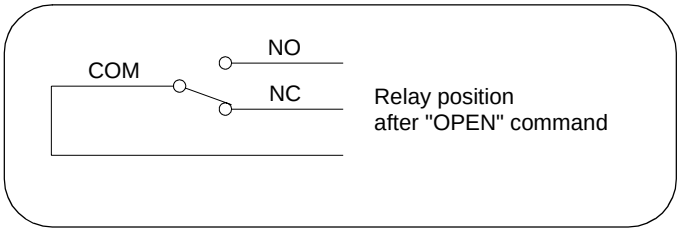
Writes to the Switch Enable Registers (base + 10_{16} through base + 16_{16}) enable you to open or close the desired channel. For example, write a 1 to bit 2 of the Switch Enable Register (base + 10_{16}) to close channel 02. Or, write a 0 to bit 15 of the register at base + 16_{16} to open channel 63.

NOTE *All relays are non-latching and will open during a power-down.*

A switch is open when contact is made between the normally closed (NC) contact and common (C). A switch is closed when contact is made between the normally open (NO) contact and common (C). Any combination of open or closed states is allowed at one time for all channels on the module.



Write a "1" to the register bit to close the relay



Write a "0" to the register bit to open the relay

Register Definitions

Manufacturer ID Register (read-only register)

Address $b+00_{16}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Manufacturer ID; Returns FFFFh = Hewlett-Packard A16 only register-based device.															

Device Type Register (read-only register)

Address $b+02_{16}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Returns 0228_{16} for the E1442A module.															

Status/Control Register

Address $b+04_{16}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write*	Undefined									D	Undefined					R
Read**	Un-def	M	Un-def	S1	Undefined				B	D	Undefined					R

***Write Bits (Status/Control Register)**		
bit 0	R	Writing a 1 to this bit resets the switch to the power-on state (all channels open). To reset, keep this bit set to 1 for a minimum of 100 ms and then set bit 0 back to a logical 0 to allow switches to be closed.
bit 6	D	Disable interrupt by writing a 1 to this bit (set back to 0 with a reset).

****READ BITS (Status/Control Register)**		
bit 0	R	A 1 at this bit resets the switch to the power-on state (all channels open). To reset, set bit 0 back to a logical 0 to allow switches to be closed.
bit 6	D	Interrupt Status: 1 = disabled, 0 = enabled.
bit 7	B	Busy Status: 1 = not busy, 0 = busy.
bit 12	S1	Fuse F4 provides +5V pull-up voltage for the NC and NO switch contacts by use of the module's internal bus (see Figures 1-8 and 1-9.) Fuse F4 status; 0 = fuse F4 not installed (factory shipped without the fuse). (A 0 also can indicate a blown fuse after installing fuse F4.) 1 = fuse F4 is installed (user must install fuse).
bit 14	M	MODID bit; if the bit is 0, the module has been selected during turn-on. Normally this bit is 1 when not in the turn-on cycle.

Switch Enable Registers

You write to the switch enable registers to close (or open) a channel. Write a "1" to the register to close a relay (channel). Write a "0" to the register to open a relay (channel). Reading any Switch Enable Register will always return $FFFF_{16}$ regardless of the channel states.

Switch Enable Register (Channels 0 - 15)

Address $b+10_{16}$	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 $FFFF_{16}$															

Switch Enable Register (Channels 16 - 31)

Address $b+12_{16}$	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 $FFFF_{16}$															

Switch Enable Register (Channels 32 - 47)

Address $b+14_{16}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Ch47	Ch46	Ch45	Ch44	Ch43	Ch42	Ch41	Ch40	Ch39	Ch38	Ch37	Ch36	Ch35	Ch34	Ch33	Ch32
Read	Always returns $FFFF_{16}$															

Switch Enable Register (Channels 48 - 63)

Address $b+16_{16}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Ch63	Ch62	Ch61	Ch60	Ch59	Ch58	Ch57	Ch56	Ch55	Ch54	Ch53	Ch52	Ch51	Ch50	Ch49	Ch48
Read	Always returns $FFFF_{16}$															

Programming Example

Beginning of Program

```
/* This program resets the E1442A, closes channels and reads the*/
/* switch's relay control registers, opens channels and scans all 64*/
/* channels on the module.                                          *

#include <sicl.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>

/* function prototypes */
void reset_sw(char *base_addr);
void delay (unsigned milliseconds);
```

Program Main

```
void main(void)
{
    double ldexp(double i, int exp);
    char *base_addr;
    int j, k;

    unsigned short chan_0_15_reg, chan_16_31_reg;  /*First 32 chan */
    unsigned short chan_32_47_reg, chan_48_63_reg; /* Sec 32 chan */
    unsigned short id_reg, dt_reg;                 /* ID and device type */
    unsigned short stat_reg;                       /* Status Register */

    /* create and open a device session */
    INST e1442a;
    e1442a = iopen("vxi,120");

    /* map the E1442A registers into user memory space */
    base_addr = imap(e1442a, I_MAP_VXIDEV, 0, 1, NULL);

    /* clear the user screen */
    clrscr();

    /* reset the E1442A */
    reset_sw(base_addr);
```

Read ID and Device Type Registers

```
/********** read the switch's ID and Device Type registers **********/

    id_reg = iwpeek((unsigned short *)(base_addr + 0x00));
    dt_reg = iwpeek((unsigned short *)(base_addr + 0x02));
    printf("ID register = 0x%4X\nDevice Type register = 0x%4X\n",
    id_reg, dt_reg);
```

Read Status Register

```
/********** read the switch's Status Register *********/

    stat_reg = iwpeek((unsigned short *)(base_addr + 0x04));
    printf("Status register = 0x%4X\n", stat_reg);
```

Close and Open Channels

```
/********** close and open channels **********/

    /* set all bits in register for channels 0-15 (offset 10) to 1 */
    iwpoke((unsigned short *)(base_addr + 0x10), 0xffff);

    /* read the E1442A relay control registers and print their value*/
    /* relay control registers always return FFFF (hex) */

    chan_0_15_reg = iwpeek((unsigned short *)(base_addr + 0x10));
    chan_16_31_reg = iwpeek((unsigned short *)(base_addr + 0x12));
    chan_32_47_reg = iwpeek((unsigned short *)(base_addr + 0x14));
    chan_48_63_reg = iwpeek((unsigned short *)(base_addr + 0x16));

    printf("Channels 00-15 register = 0x%4X\n", chan_0_15_reg);
    printf("Channels 16-31 register = 0x%4X\n", chan_16_31_reg);
    printf("Channels 32-47 register = 0x%4X\n", chan_32_47_reg);
    printf("Channels 48-63 register = 0x%4X\n", chan_48_63_reg);

    delay (100);    /* wait 100 milliseconds before resetting module */

    /* reset the E1442A to open all closed channels */
    /* writing a 0 to the channels registers will also open channels */
    reset_sw(base_addr);
```

Scan Channels

```
/********** scanning channels **********/

    /* scan channels 0-15 (register offset 10) */
    for (k=0; k15; k++)
    {
    iwpoke ((unsigned short *)(base_addr + 0x10), ldexp(1,k));
    delay (50);   /* delay to allow mechanical relays to close*/
    }
    /* sets all bits to 0 to open last closed channel */
    iwpoke ((unsigned short *)(base_addr + 0x10), 0);

    /* scan channels 16-31 (register offset 12) */
    for (k=0; k15; k++)
    {
    iwpoke ((unsigned short *)(base_addr + 0x12), ldexp(1,k));
    delay (50);
    }
    /* sets all bits to 0 to open last closed channel */
    iwpoke ((unsigned short *)(base_addr + 0x12), 0);

/* scan channels 32-47 (register offset 14) */
    for (k=0; k15; k++)
    {
    iwpoke ((unsigned short *)(base_addr + 0x14), ldexp(1,k));
    delay (50);
    }
    /* set all bits to 0 to open last closed channel */
    iwpoke ((unsigned short *)(base_addr + 0x14), 0);
```

```
    /* scan channels 48-63 (register offset 16) */
    for (k=0; k15; k++)
    {
    iwpoke ((unsigned short *)(base_addr + 0x16), ldexp(1,k));
    delay (50);
    }
    /* set all bits to 0 to open last closed channel */
    iwpoke ((unsigned short *)(base_addr + 0x16), 0);

    /* close session */
    iclose(e1442a);


}   /* end of main */
```

Reset Function

```
/*********************************************************/

void reset_sw(char *base_addr)
    /* reset the module; open all relays (write a 1 to status bit 0)  */
    /* delay 100 ms for reset then set bit to 0 to allow closing of   */
    /* switches*/
}
    /* this function resets the switch module */
       iwpoke((unsigned short *)(base_addr + 0x04), 1);
       delay (100);
       iwpoke((unsigned short *)(base_addr + 0x04), 0); }
}
```

Appendix C
E1442A Error Messages

Error Types

Table C-2 lists the error messages generated by the E1442A Form C Switch module firmware when programmed by SCPI. Errors with negative values are governed by the SCPI standard and are categorized in Table C-1. Error numbers with positive values are not governed by the SCPI standard. See the *E1406 Command Module User's Manual* for further details on these errors.

Table C-1. Error Types

Range	Error Types Description
-199 to -100	Command Errors (syntax and parameter errors).
-299 to -200	Execution Errors (instrument driver detected errors)
-399 to -300	Device Specific Errors (instrument driver errors that are not command nor execution errors).
-499 to -400	Query Errors (problem in querying an instrument)

Error Messages

Table C-2. E1442A Error Messages

Code	Error Message	Potential Cause(s)
-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.
-222	Data out of range	Parameter value is outside valid range.
-224	Illegal parameter value	Attempting to execute a command with a parameter not applicable to the command.
-240	Hardware error	Command failed due to hardware problem.
-310	System error, Internal driver error.	This error can result if an excessively long parameter list is entered.
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.
1510	Trigger source non-existent	Selected trigger source is not available on this platform (e.g., some triggers are not available on VXI B-size mainframes).
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 (e.g.,, 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 a scan without the INIT command.
2009	Too many channels in channel list	Attempting to address more channels than available in the switchbox.
2010	Scan mode not allowed on this card	The selected scanning mode is not allowed with this module or you have misspelled the mode parameter (see SCAN:MODE command).
2011	Empty channel list	No valid channels are specified in the <*channel_list*>.
2012	Invalid Channel Range	Invalid channel(s) specified in SCAN <*channel_list*> command. Attempting to begin scanning when no valid channel list is defined.
2017	Config error 17, Slot 0 functions disabled	Attempt to run a downloaded scan list with ARM:COUNt set to a value other than 1. Applies to FET switches only.
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.
2601	Channel list required	Sending a command requiring a channel list without the channel list.

Symbols

A

B

C

C (continued)

D

E

E (continued)

I

L

O

P

R

S

S (continued)

T

W

Notes:

Agilent Technologies

Manual Part Number: E1442-90003
Printed in U.S.A. E1000