bus

Agilent 75000 Series C

Agilent E1465A/E1466A/E1467A Relay Matrix Switch Modules

User's Manual and SCPI Programming Guide

Where to Find It - Online and Printed Information:

If your instrument has a Agilent VXI*plug&play* driver, product information is organized as indicated below.

System installation (hardware/software)	VXIbus Configuration Guide* Agilent VIC (VXI installation software)*
Module configuration and wiring	This Manual
SCPI programming	This Manual
SCPI example programs	This Manual
SCPI command reference	This Manual
Register-based programming	This Manual
VXI*plug&play* programming	VXI*plug&play* Online Help
VXI*plug&play* example programs	VXI*plug&play* Online Help
VXI*plug&play* function reference	VXI*plug&play* Online Help
Soft Front Panel information	VXI*plug&play* Online Help
VISA language information	Agilent VISA User's Guide
Agilent VEE programming information	Agilent VEE User's Manual

**Supplied with Agilent Command Modules,Embedded Controllers, and VXLink.*

Manual Part Number: E1466-90006
Printed in Malaysia E0406

Contents

Agilent E1465/66/67A Relay Matrix Switch Modules

Notes

Certification

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

Warranty

This Agilent Technologies product is warranted against defects in materials and workmanship for a period of one (1) year from date of shipment. Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other Agilent products. During the warranty period, Agilent Technologies will, at its option, either repair or replace products which prove to be defective.

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

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

Limitation Of Warranty

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

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

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

Exclusive Remedies

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

Notice

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

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 E1465A/E1466A/E1467A Relay Matrix Switch Module User's Manual
Edition 6 Rev 2

Printing History

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

Edition 1	July 1991
Edition 2	July 1993
Edition 3	June 1995
Edition 4	January 1996
Edition 5	May 1996
Edition 6 (Part Number E1466-90006)	November 1996
Edition 6 Rev 2 (Part Number E1466-90006)	April 2006

Safety Symbols

⚠	Instruction manual symbol affixed to product. Indicates that the user must refer to the manual for specific WARNING or CAUTION information to avoid personal injury or damage to the product.
⏚	Indicates the field wiring terminal that must be connected to earth ground before operating the equipment—protects against electrical shock in case of fault.
⏊ or ⊥	Frame or chassis ground terminal—typically connects to the equipment's metal frame.
∿	Alternating current (AC).
⎓	Direct current (DC).
⚡	Indicates hazardous voltages.
WARNING	Calls attention to a procedure, practice, or condition that could cause bodily injury or death.
CAUTION	Calls attention to a procedure, practice, or condition that could possibly cause damage to equipment or permanent loss of data.

WARNINGS

The following general safety precautions must be observed during all phases of operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the product. 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 an Agilent Technologies Sales and Service Office for service and repair to ensure that safety features are maintained.

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

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

DECLARATION OF CONFORMITY

According to ISO/IEC Guide 22 and CEN/CENELEC EN 45014

Manufacturer's Name: Agilent Technologies, Incorporated

Manufacturer's Address:
Measurement Product Generation Unit
815 14th ST. S.W.
Loveland, CO 80537 USA

Declares, that the product

Product Name: Relay Matrix Switch
Model Number: E1465A/E1466A/E1467A
Product Options: *This declaration covers 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:

EMC	Standard	Limit
	IEC 61326-1:1997+A1:1998 / EN 61326-1:1997+A1:1998	
	CISPR 11:1997 +A1:1997 / EN 55011:1998	*Group 1 Class A* [1]
	IEC 61000-4-2:1995+A1:1998 / EN 61000-4-2:1995	*4kV CD, 8kV 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	*0.5kV signal lines, 1kV 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	*3V, 0.15-80 MHz*
	IEC 61000-4-11:1994 / EN 61000-4-11:1994	*I 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 3111-1:1994

Supplemental Information:

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

September 5, 2000
Date

Jim White
Name

Quality Manager
Title

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

Notes

Notes

Notes

Chapter 1
Getting Started

Using This Chapter

This chapter includes a relay matrix description and addressing guidelines. Chapter contents are as follows:

Matrix Description

This manual supports the Agilent E1465A, E1466A, and E1467A Relay Matrix Switch Modules. These switches are VXIbus C-size register-based modules that can operate with an Agilent command module (e.g., Agilent E1406A). Four 4 x 16 submatrices are implemented on the main PC board with 256 latching relays. Terminal modules convert the submatrices into the 4 x 64 (Agilent E1466A), 8 x 32 (Agilent E1467A), or the 16 x 16 (Agilent E1465A) matrices. **The matrix model number is determined by the terminal module connected to the PC board.**

Note If no terminal module is connected, the relay matrix switch module will default to an E1466A. In order to program the E1465A and E1467A, make certain the terminal module is connected.

The Agilent E1465A Relay Matrix Switch provides a 16 x 16 two-wire crosspoint matrix. This 16 x 16 matrix is created by connecting the terminal module. The terminal module connects the columns of the submatrices of A, B, C, and D. Figure 1-1 shows a simplified diagram of the E1465A module.

The Agilent E1466A Relay Matrix Switch provides a 4 x 64 two-wire crosspoint matrix. This 4 x 64 matrix is created by connecting the terminal module. The terminal module connects the rows of submatrices A, B, C, and D. Figure 1-2 shows a simplified diagram of the E1466A module.

The Agilent E1467A Relay Matrix Switch provides an 8 x 32 two-wire crosspoint matrix. This 8 x 32 matrix is created by connecting the terminal module. The terminal module connects the rows of submatrices A and C, and rows of submatrices B and D. The columns of submatrices A and B, and columns of submatrices C and D are also connected. Figure 1-3 shows a simplified diagram of the E1467A module.

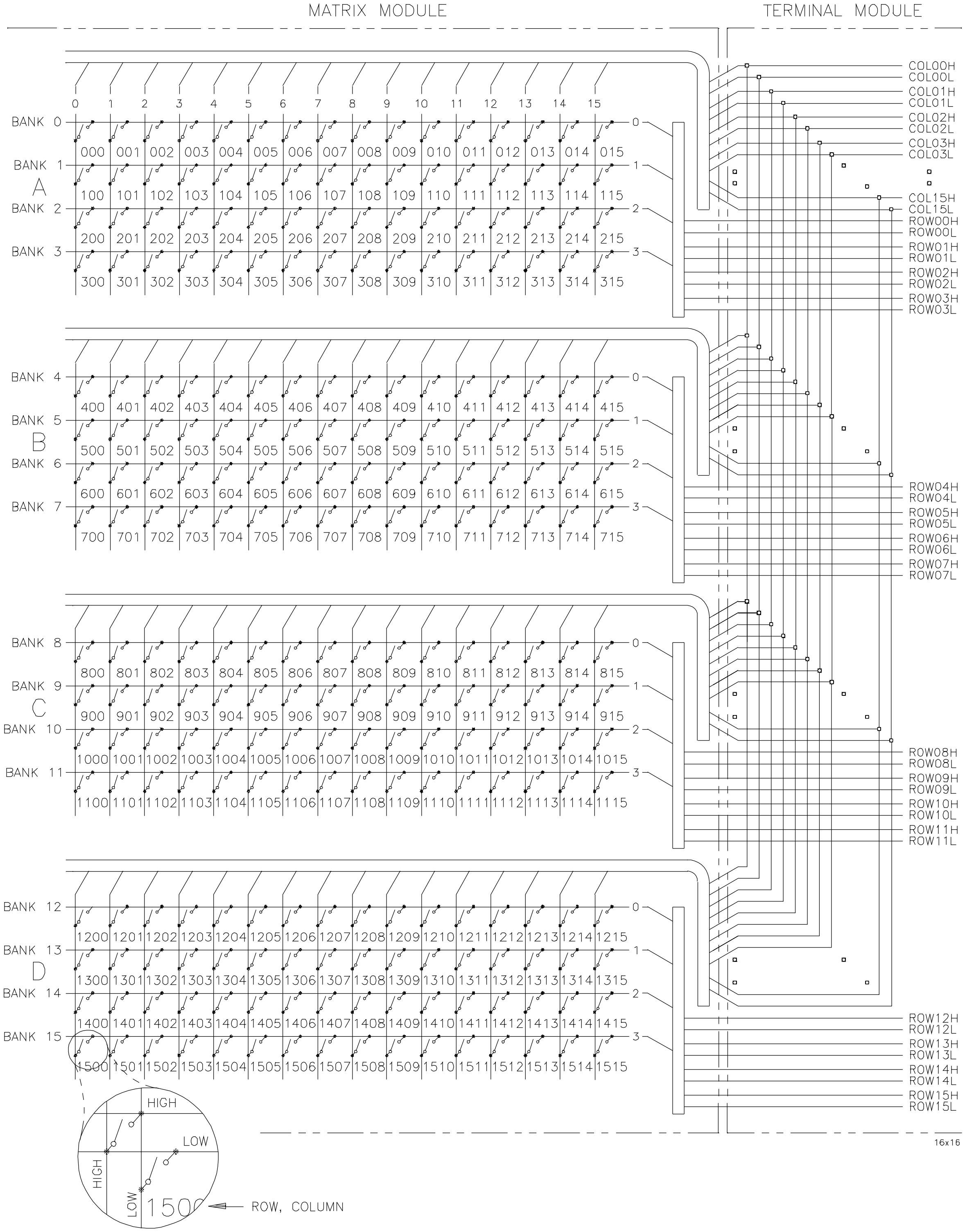


Figure 1-1. The E1465A 16 x 16 Matrix Module

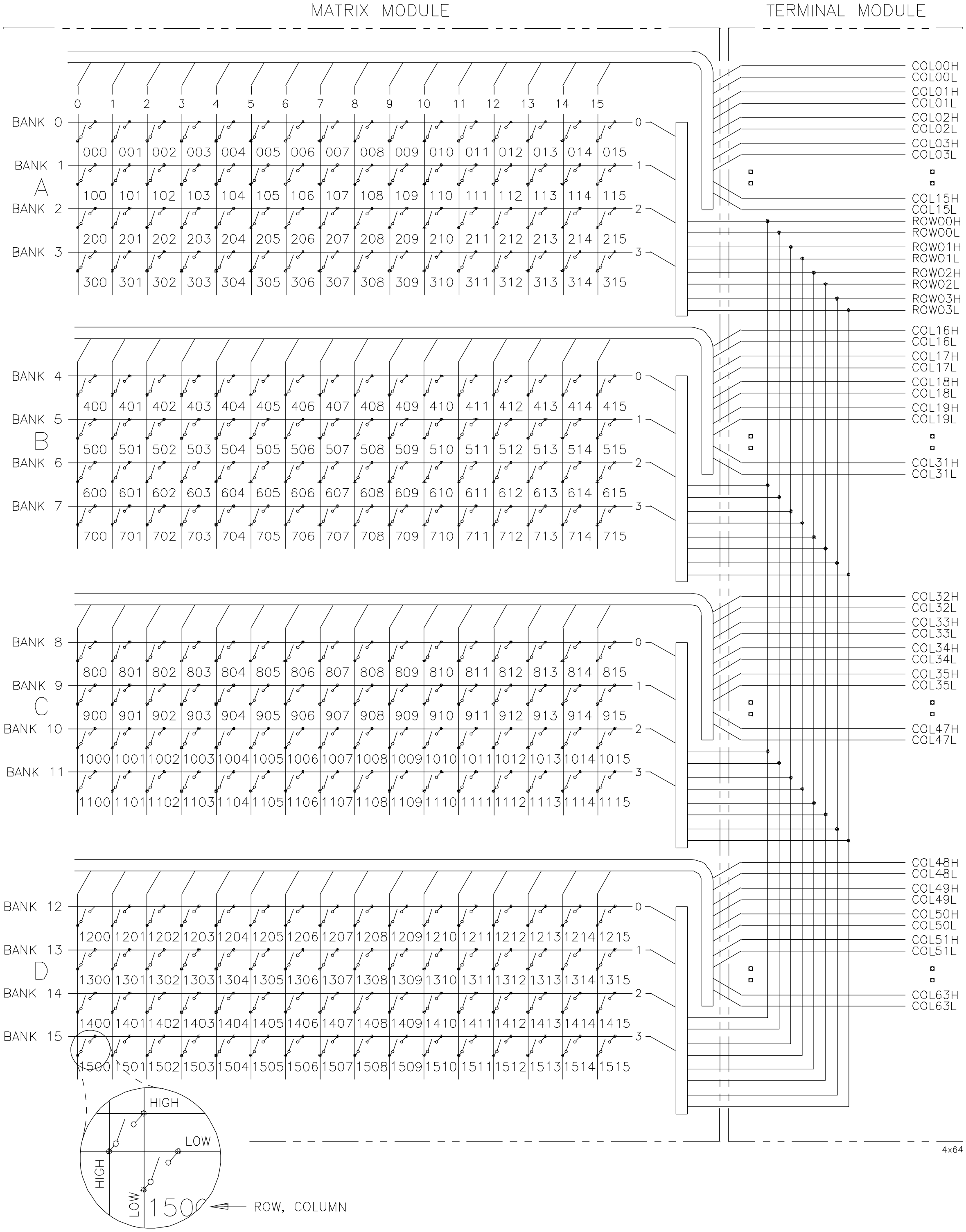


Figure 1-2. The E1466A 4 x 64 Matrix Module

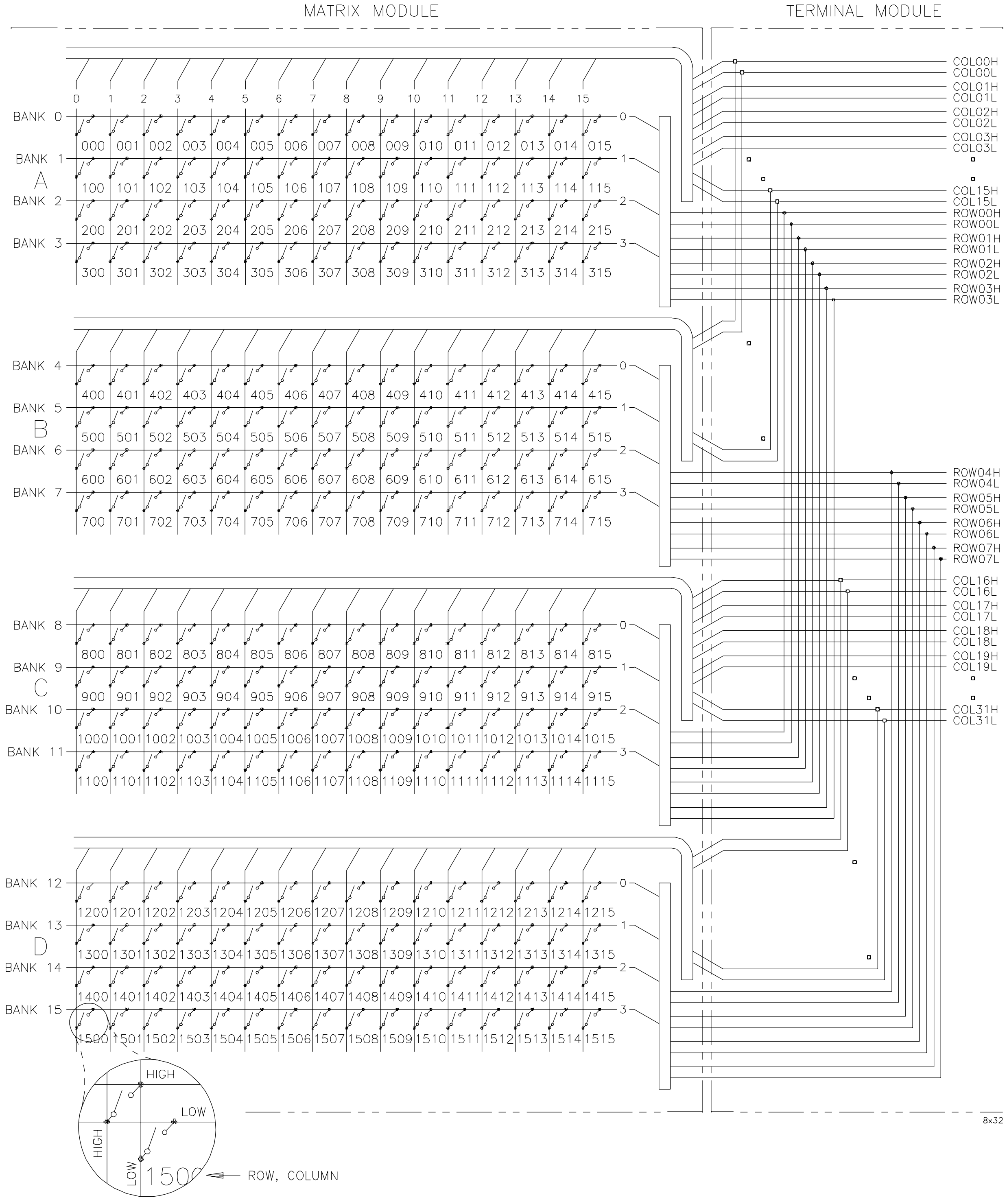


Figure 1-3. The E1467A 8 x 32 Matrix Module

Instrument Definition

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

Programming the Matrix

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

Another way to program the matrix module is to use an Agilent command module and SCPI (Standard Commands for Programmable Instruments) commands. With SCPI commands the Agilent command module parses the commands and writes to the appropriate relay module register. The examples in this manual use the SCPI programming language. See Appendix B, "Matrix Registers" for examples on writing directly to the registers.

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

- HP 9000 Series 200/300 computer running BASIC
- HP Vectra computer (or compatible) with an Agilent 82335A GPIB Interface card (with command library) running Borland® Turbo C
- Agilent E1499A V/382 computer running BASIC/UNIX, C (This configuration is used in Appendix B for register-based programming).

See the *C-Size VXIbus Systems Configuration Guide* for information on additional configurations.

Specifying SCPI Commands

To address specific channels (relays) within a relay matrix, you must specify the SCPI command and matrix channel list. The following are the most commonly used SCPI commands:

Table 1-1. SCPI Commands

SCPI Commands	Command Description
CLOSe *<channel_list>*	Closes the relays specified
OPEN *<channel_list>*	Opens the relays specified
SCAN *<channel_list>*	Closes the relays specified, one at a time

The Channel List

The channel list is a combination of the card number and the channel numbers. The channel list takes the form of @ssrrcc where

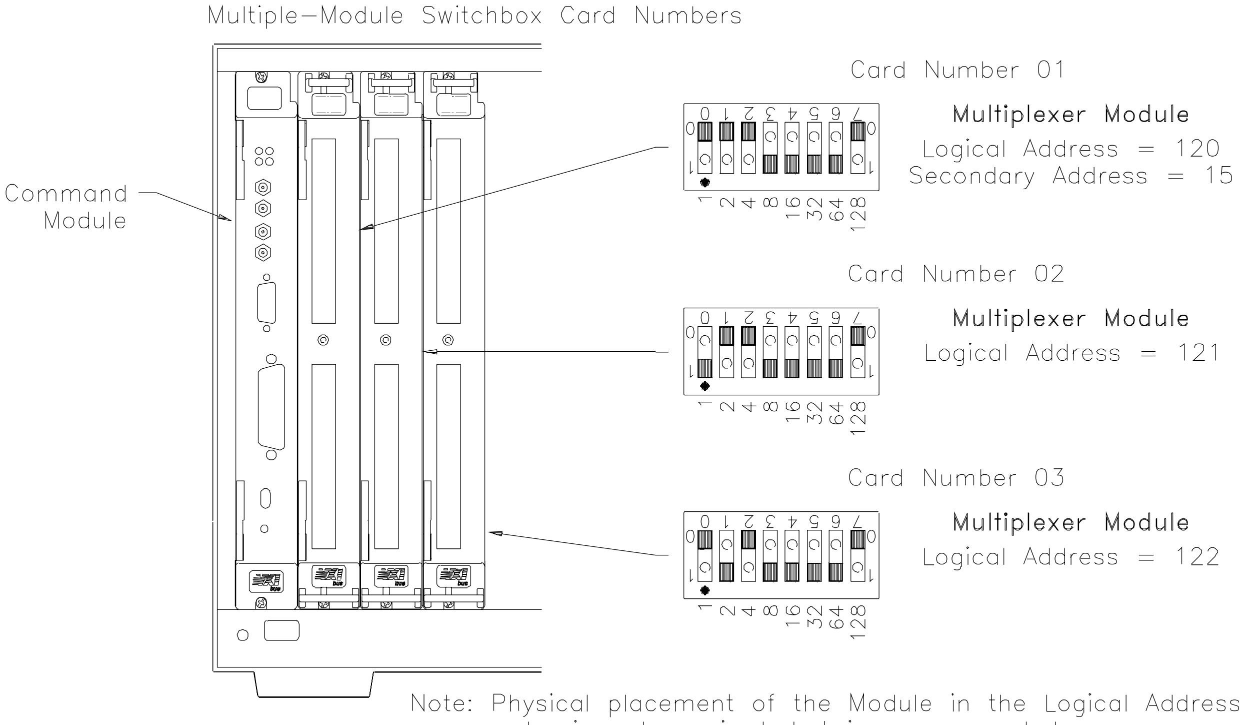
ss = matrix card number (00-99)
rr = row number of the matrix
cc = column number of the matrix

The Card Number

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

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

- Multiple-Module Switchbox. In a multiple-module switchbox configuration, multiplexer modules are set to successive logical addresses. The multiplexer module with the lowest logical address is always card number 01. The card number with the next successive logical address is 02, and so on. Figure 1-4 illustrates the card numbers and logical addresses of a typical multiple-module switchbox configuration. In Chapter 2, an example of addressing a switchbox configuration is shown in the "Using Multiple Mainframes" section.



Note: Physical placement of the Module in the Logical Address order is not required, but is recommended.

Figure 1-4. Card Numbers in a Multiple-Module Switchbox

The Channel Addresses

The channel address is the **rrcc** of the channel list. This address determines which relay will be addressed. Use a comma (**,**) to form a channel list or use a colon (**:**) to form a channel range. You can address the following:

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

Table 1-2. Matrix Channel Numbers

Matrix Module	Rows (rr)	Columns (cc)
E1465A 16 x 16 Relay Matrix	00 - 15	00 - 15
E1466A 4 x 64 Relay Matrix	00 - 03	00 - 63
E1467A 8 x 32 Relay Matrix	00 - 07	00 - 31

Only valid channels can be accessed in a channel list or channel range. Also, the channel range must be from a lower channel number to a higher channel number. For example, CLOS (@10000:20303) is acceptable, but CLOS (@20303:10000) generates an error.

Initial Operation

Two example programs follow which use BASIC and TURBO C languages to get you started using the relay matrix switch module. The first example assumes an HP 9000 Series 200/300 controller and a General Purpose Interface Bus (GPIB).[1] The second example assumes an HP Vectra PC (or compatible) with an Agilent 82335A GPIB Interface card (with command library) running Borland® Turbo C.

The program closes row 03, column 12 of an Agilent E1465A 16 x 16 Relay Matrix at logical address 120 (secondary address = 120/8 = 15) and queries the result. The result is returned to the controller and displayed (1 = relay closed, 0 = relay open). See Chapter 5 for information on the SCPI commands.

BASIC

```
10 OUTPUT 70915; "*RST"               ! Resets the module.
20 OUTPUT 70915; "CLOS  (@10312)"     ! Closes row 03, column 12 on
                                      module number 1.
30 OUTPUT 70915; "CLOS? (@10312)"     ! Query channel 10312.
40 ENTER 70915; Value                 ! Enter result into variable
                                      Value.
50 PRINT Value                        ! Print results (should print "1"
                                      to indicate that the channel is
                                      closed).
60 END                                ! Terminate program.
```

TURBO C

```
#include <stdio.h>
#include <chpib.h>                          /*Include file for GPIB*/

#define ISC 7L
#define MATRIX 70915L                  /*Matrix default address*/
#define TASK1 "*RST"                   /*Command for a reset*/
#define TASK2 "CLOS (@10312)"          /*Command to close row 3, column 12*/
#define TASK3 "CLOS? (@10312)"         /*Command to query row 3, column 12*/
```

Continued on next page.

[1] (GPIB is the implementation of IEEE Std 488.1-1987)

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

                                        /*Output commands to matrix*/
     error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");
     error_handler (IOOUTPUTS (MATRIX, TASK1, 4), "OUTPUT command");
     error_handler (IOOUTPUTS (MATRIX, TASK2, 15), "OUTPUT command");
     error_handler (IOOUTPUTS (MATRIX, TASK3, 15), "OUTPUT command");

                                        /*Enter from matrix*/

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

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

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

Notes

Chapter 2
Configuring the Matrix Modules

Using This Chapter

This chapter shows how to connect external wiring to the matrix modules and how to connect multiple modules together to form larger matrices. This chapter contains the following sections:

WARNINGS and CAUTIONS

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

CAUTION **MAXIMUM INPUTS. The maximum voltage that can be applied to any terminal is 200 Vdc/170 Vrms . The maximum current that can be applied to any row or column is 1 A dc or ac peak. The maximum power that can be applied to any terminal is 30 W or 62.5 VA (resistive).**

STATIC ELECTRICITY. Static electricity is a major cause of component failure. To prevent damage to the electrical components in the matrix module, observe anti-static techniques when removing or installing the module or when working on the module.

Setting the Module Address Switch

The logical address switch (LADDR) factory setting is 120. Valid address values are from 1 to 255. The matrix module can be configured as a single instrument or as a switchbox. If the matrix module is used with an Agilent E1406A Command Module in a C-size mainframe, refer to the *C-Size VXIbus Systems Configuration Guide* for addressing information. Refer to Figure 2-1 for switch position information.

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

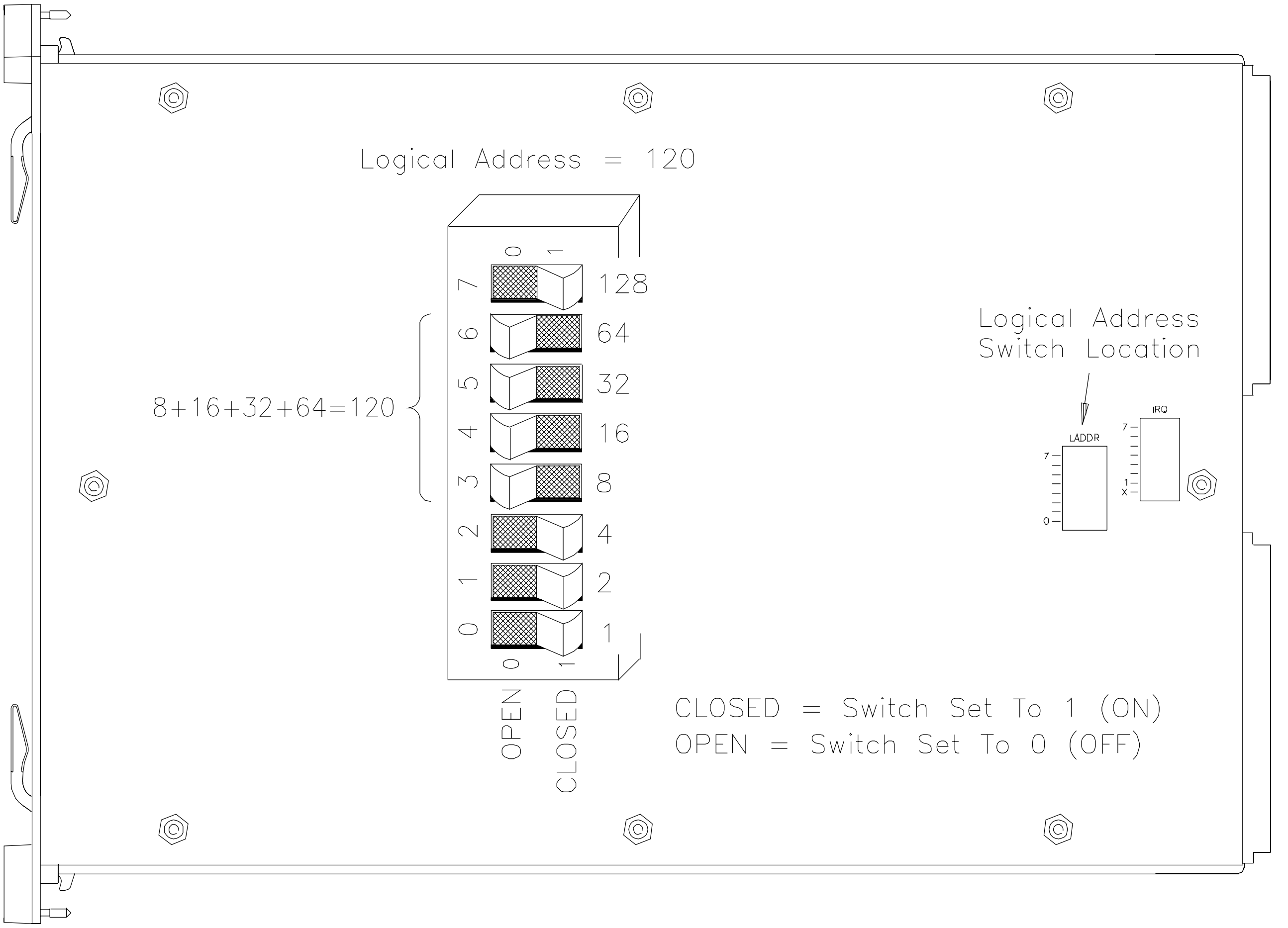


Figure 2-1. Selecting the Module Address

Selecting the Interrupt Line

The matrix module generates an interrupt after a channel has been closed. These interrupts are sent to, and acknowledgements are received from, the command module (e.g., Agilent E1406A) via the VXIbus backplane interrupt lines. For applications where the matrix module is installed in an Agilent 75000 Series C mainframe and is a servant of the command module, the interrupt line jumper does not have to be moved. Refer to Figure 2-2 to change the interrupt line.

You can select seven different interrupt line levels. Line X disables the interrupt and should not be used. The module's factory setting is line 1. To change, remove the four-pin jumper (Agilent part number 1258-0247) from the old line location and reinstall in the new line location. If you are setting the interrupt line to something other than 1, see the *Agilent E1406 Command Module User's Manual* for additional information. If the four-pin jumper is not used, the two jumper locations must have the same interrupt line selected.

Note When the Agilent E1405/06 Command Module is the resource manager, the interrupt line jumper *must* be installed in position 1. However, if you are using an Agilent E1499A (Agilent V/382) with the Agilent E1405/06 Command Module, interrupt line 2 should be selected. Level X interrupt line should not be used under normal operating conditions. See the *C-Size VXIbus Systems Configuration Guide* for additional information.

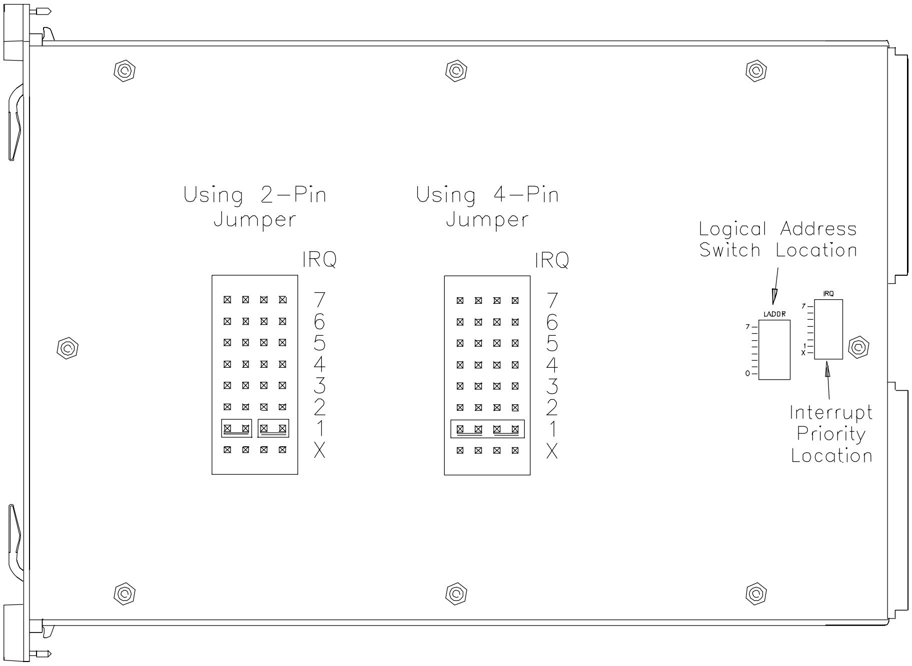
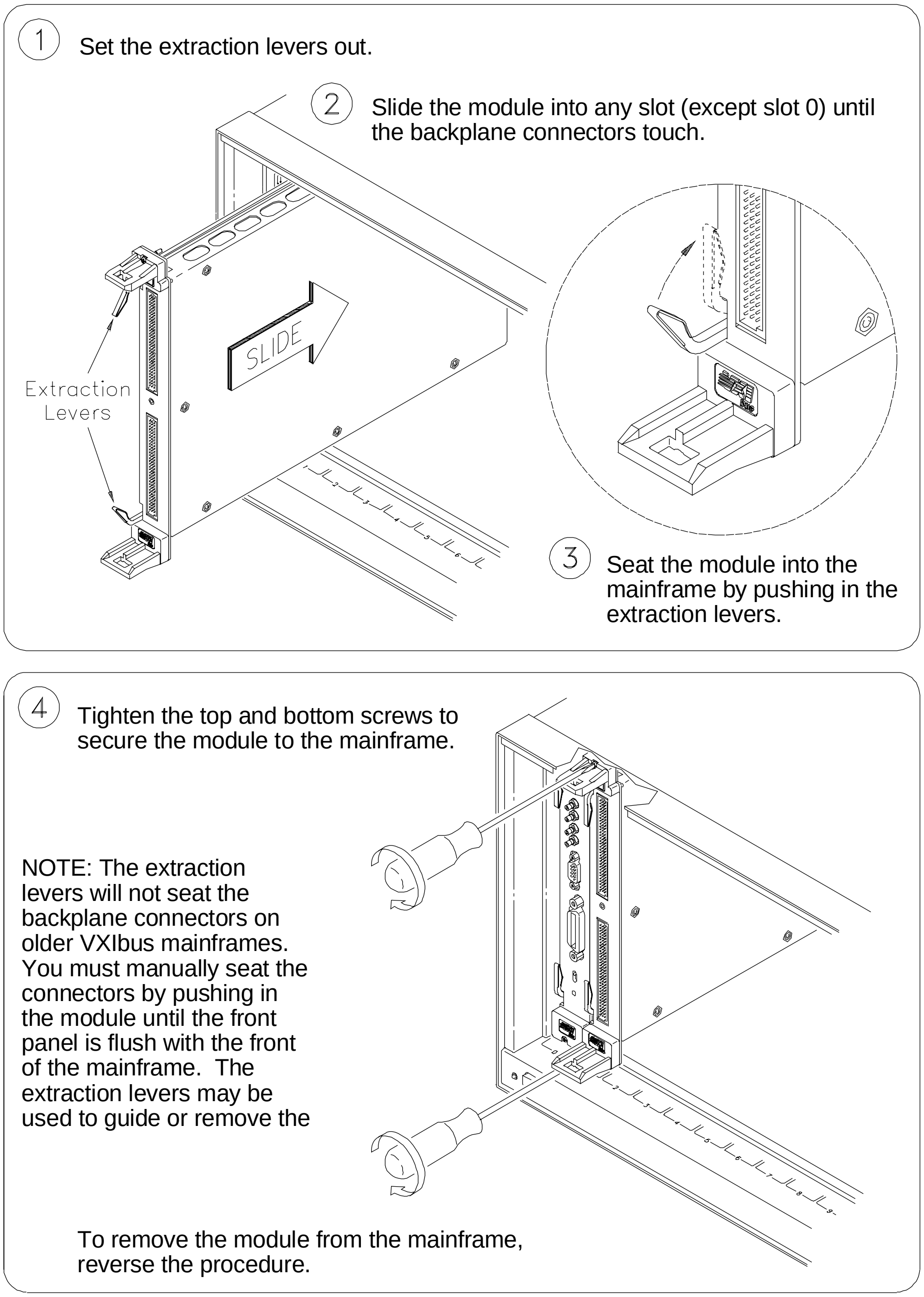


Figure 2-2. Selecting the Interrupt Line

Installing the Relay Matrix Switch Module in a Mainframe

The Agilent E1465/66/67A may be installed in any slot (except slot 0) in a C-size VXIbus mainframe. Refer to Figure 2-3 to install the module in a mainframe.



1 Set the extraction levers out.

2 Slide the module into any slot (except slot 0) until the backplane connectors touch.

3 Seat the module into the mainframe by pushing in the extraction levers.

4 Tighten the top and bottom screws to secure the module to the mainframe.

NOTE: The extraction levers will not seat the backplane connectors on older VXIbus mainframes. You must manually seat the connectors by pushing in the module until the front panel is flush with the front of the mainframe. The extraction levers may be used to guide or remove the

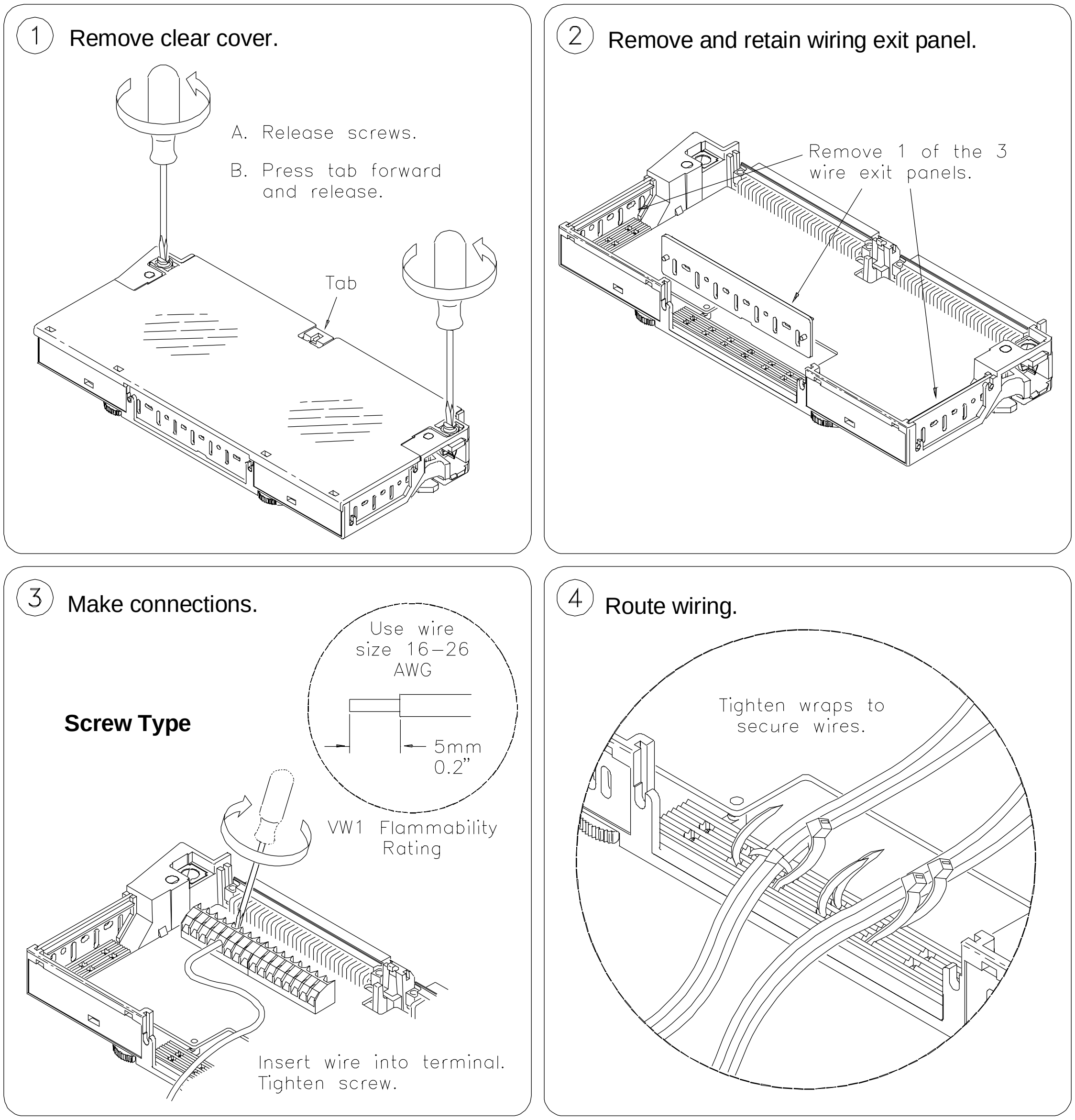
To remove the module from the mainframe, reverse the procedure.

Figure 2-3. Installing the Switch Module in a VXIbus Mainframe

Wiring the Standard Terminal Module

User wiring to the matrix modules are to the High (H) and Low (L) terminal connections. Figures 2-4 and 2-5 show the rudiments of the terminal module assembly. Expansion connectors allow for creating larger matrices (see later in this chapter). For information on the E1467A Option 211 Matrix Expansion Terminal Module, see the section dedicated to that product later in this chapter.

Maximum terminal wire size is No. 16 AWG. Wire ends should be stripped 6 mm (0.25 in.) and tinned. When wiring all channels, use a smaller gauge wire (No. 20 - 22 AWG).



1 Remove clear cover.

2 Remove and retain wiring exit panel.

3 Make connections.

4 Route wiring.

A. Release screws.
B. Press tab forward and release.
Tab
Remove 1 of the 3 wire exit panels.
Screw Type
Use wire size 16–26 AWG
5mm 0.2"
VW1 Flammability Rating
Insert wire into terminal. Tighten screw.
Tighten wraps to secure wires.

Figure 2-4. Wiring the Standard Terminal Module

Continued on next page.

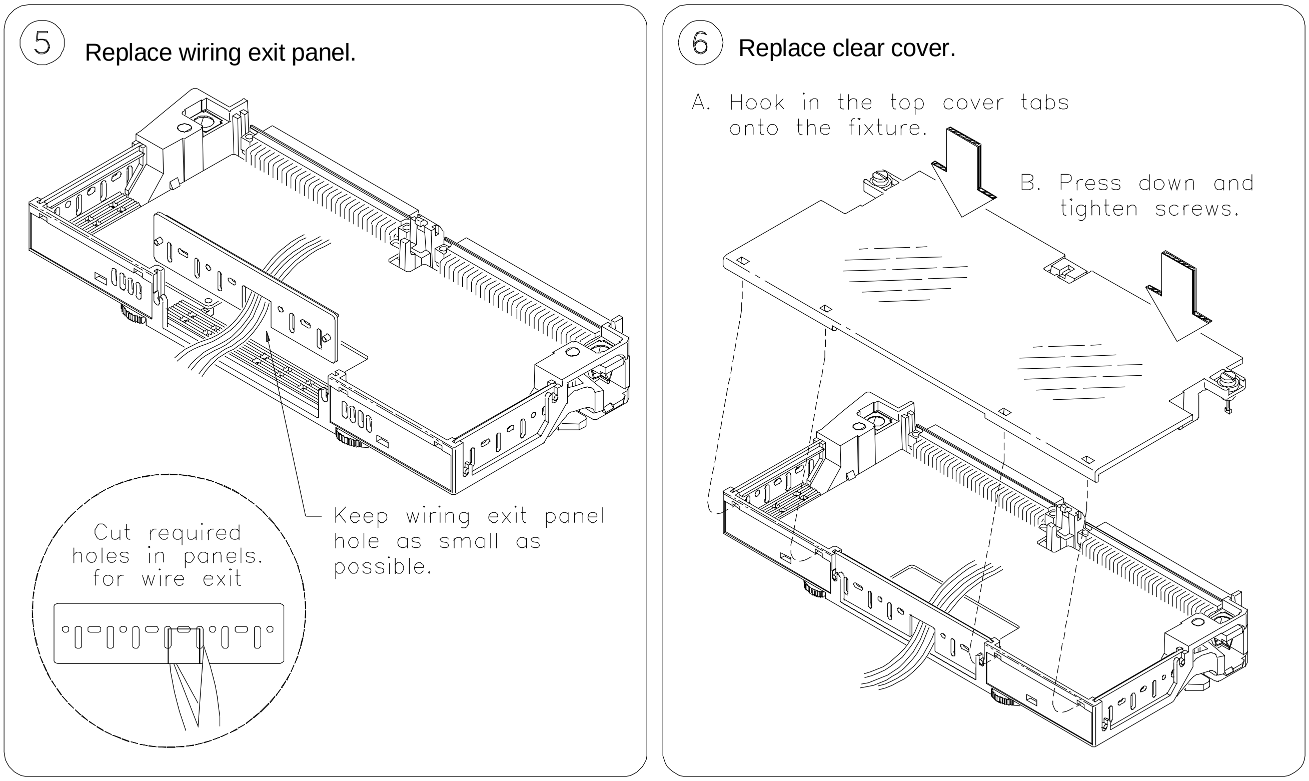
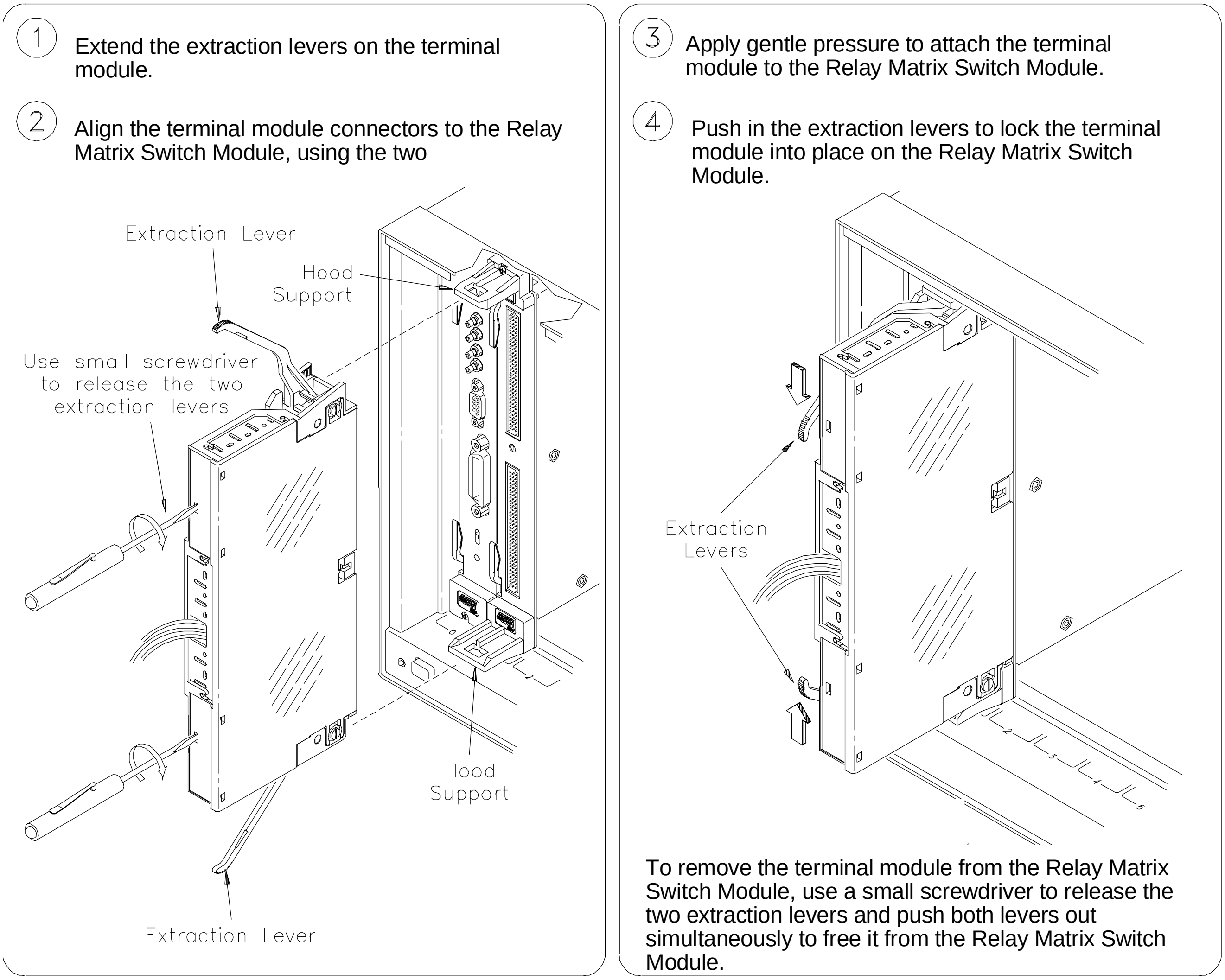

5
Replace wiring exit panel.
Keep wiring exit panel hole as small as possible.
Cut required holes in panels. for wire exit
6
Replace clear cover.
A. Hook in the top cover tabs onto the fixture.
B. Press down and tighten screws.

Figure 2-5. Wiring the Standard Terminal Module

Continued from previous page.

Attaching the Standard Terminal Module to the Relay Matrix Switch Module

Figure 2-6 shows how to install the standard terminal modules in the relay matrix switch module. For information on installing the Agilent E1467A Option 211 Matrix Expansion Terminal Module, see the section dedicated to that product later in this chapter.



1 Extend the extraction levers on the terminal module.

2 Align the terminal module connectors to the Relay Matrix Switch Module, using the two

3 Apply gentle pressure to attach the terminal module to the Relay Matrix Switch Module.

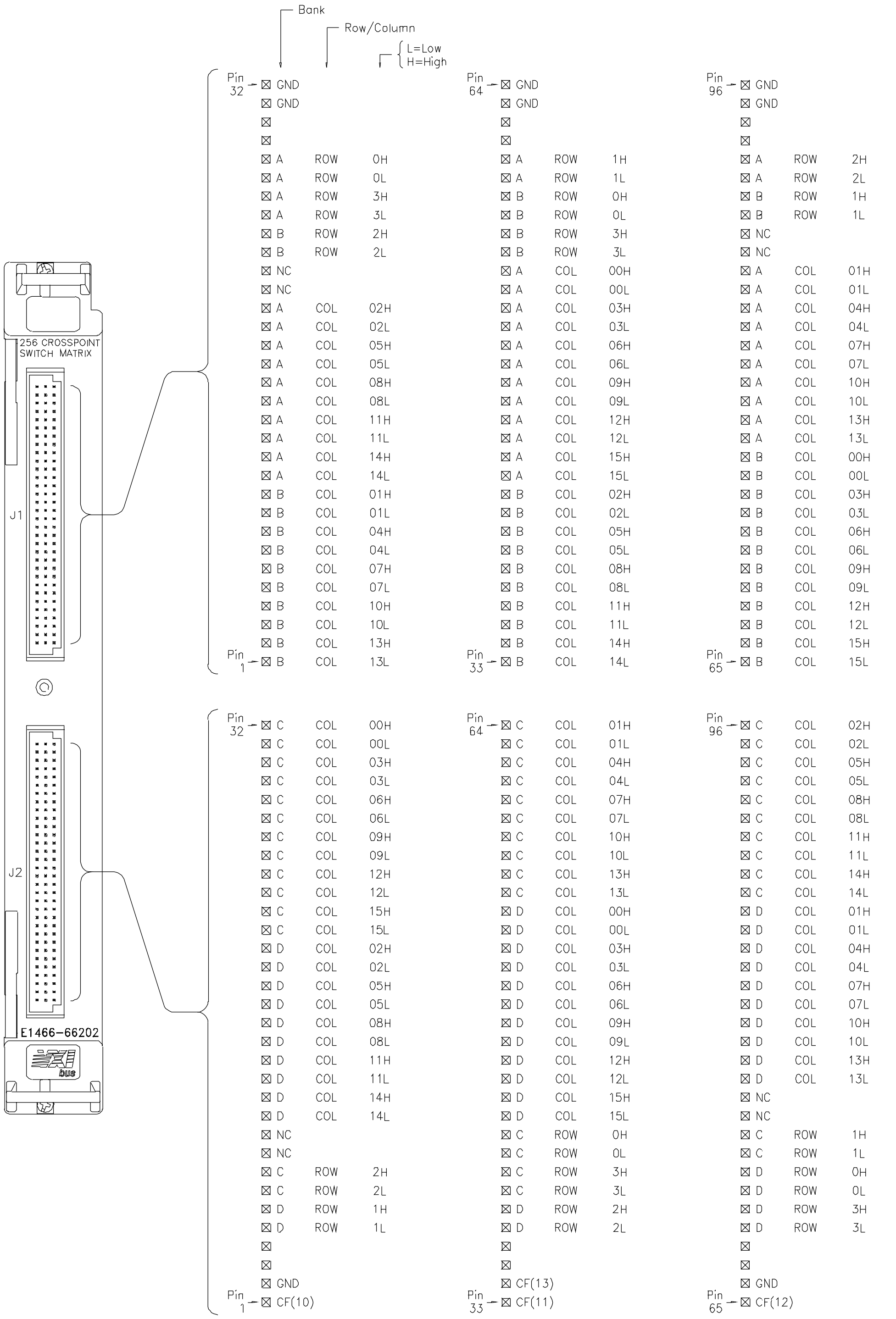
4 Push in the extraction levers to lock the terminal module into place on the Relay Matrix Switch Module.

To remove the terminal module from the Relay Matrix Switch Module, use a small screwdriver to release the two extraction levers and push both levers out simultaneously to free it from the Relay Matrix Switch Module.

Figure 2-6. Attaching the Standard Terminal Module to the Relay Matrix Switch Module

Connector Pin-Out

Figure 2-7 shows the front panel of the Agilent E1465/66/67A and the connector pin-out that mates to the terminal module.



Pin 32	☒ GND			Pin 64	☒ GND			Pin 96	☒ GND		
	☒ GND				☒ GND				☒ GND		
	☒				☒				☒		
	☒				☒				☒		
	☒ A	ROW	0H		☒ A	ROW	1H		☒ A	ROW	2H
	☒ A	ROW	0L		☒ A	ROW	1L		☒ A	ROW	2L
	☒ A	ROW	3H		☒ B	ROW	0H		☒ B	ROW	1H
	☒ A	ROW	3L		☒ B	ROW	0L		☒ B	ROW	1L
	☒ B	ROW	2H		☒ B	ROW	3H		☒ NC		
	☒ B	ROW	2L		☒ B	ROW	3L		☒ NC		
	☒ NC				☒ A	COL	00H		☒ A	COL	01H
	☒ NC				☒ A	COL	00L		☒ A	COL	01L
	☒ A	COL	02H		☒ A	COL	03H		☒ A	COL	04H
	☒ A	COL	02L		☒ A	COL	03L		☒ A	COL	04L
	☒ A	COL	05H		☒ A	COL	06H		☒ A	COL	07H
	☒ A	COL	05L		☒ A	COL	06L		☒ A	COL	07L
	☒ A	COL	08H		☒ A	COL	09H		☒ A	COL	10H
	☒ A	COL	08L		☒ A	COL	09L		☒ A	COL	10L
	☒ A	COL	11H		☒ A	COL	12H		☒ A	COL	13H
	☒ A	COL	11L		☒ A	COL	12L		☒ A	COL	13L
	☒ A	COL	14H		☒ A	COL	15H		☒ B	COL	00H
	☒ A	COL	14L		☒ A	COL	15L		☒ B	COL	00L
	☒ B	COL	01H		☒ B	COL	02H		☒ B	COL	03H
	☒ B	COL	01L		☒ B	COL	02L		☒ B	COL	03L
	☒ B	COL	04H		☒ B	COL	05H		☒ B	COL	06H
	☒ B	COL	04L		☒ B	COL	05L		☒ B	COL	06L
	☒ B	COL	07H		☒ B	COL	08H		☒ B	COL	09H
	☒ B	COL	07L		☒ B	COL	08L		☒ B	COL	09L
	☒ B	COL	10H		☒ B	COL	11H		☒ B	COL	12H
	☒ B	COL	10L		☒ B	COL	11L		☒ B	COL	12L
	☒ B	COL	13H		☒ B	COL	14H		☒ B	COL	15H
Pin 1	☒ B	COL	13L	Pin 33	☒ B	COL	14L	Pin 65	☒ B	COL	15L

Pin 32	☒ C	COL	00H	Pin 64	☒ C	COL	01H	Pin 96	☒ C	COL	02H
	☒ C	COL	00L		☒ C	COL	01L		☒ C	COL	02L
	☒ C	COL	03H		☒ C	COL	04H		☒ C	COL	05H
	☒ C	COL	03L		☒ C	COL	04L		☒ C	COL	05L
	☒ C	COL	06H		☒ C	COL	07H		☒ C	COL	08H
	☒ C	COL	06L		☒ C	COL	07L		☒ C	COL	08L
	☒ C	COL	09H		☒ C	COL	10H		☒ C	COL	11H
	☒ C	COL	09L		☒ C	COL	10L		☒ C	COL	11L
	☒ C	COL	12H		☒ C	COL	13H		☒ C	COL	14H
	☒ C	COL	12L		☒ C	COL	13L		☒ C	COL	14L
	☒ C	COL	15H		☒ D	COL	00H		☒ D	COL	01H
	☒ C	COL	15L		☒ D	COL	00L		☒ D	COL	01L
	☒ D	COL	02H		☒ D	COL	03H		☒ D	COL	04H
	☒ D	COL	02L		☒ D	COL	03L		☒ D	COL	04L
	☒ D	COL	05H		☒ D	COL	06H		☒ D	COL	07H
	☒ D	COL	05L		☒ D	COL	06L		☒ D	COL	07L
	☒ D	COL	08H		☒ D	COL	09H		☒ D	COL	10H
	☒ D	COL	08L		☒ D	COL	09L		☒ D	COL	10L
	☒ D	COL	11H		☒ D	COL	12H		☒ D	COL	13H
	☒ D	COL	11L		☒ D	COL	12L		☒ D	COL	13L
	☒ D	COL	14H		☒ D	COL	15H		☒ NC		
	☒ D	COL	14L		☒ D	COL	15L		☒ NC		
	☒ NC				☒ C	ROW	0H		☒ C	ROW	1H
	☒ NC				☒ C	ROW	0L		☒ C	ROW	1L
	☒ C	ROW	2H		☒ C	ROW	3H		☒ D	ROW	0H
	☒ C	ROW	2L		☒ C	ROW	3L		☒ D	ROW	0L
	☒ D	ROW	1H		☒ D	ROW	2H		☒ D	ROW	3H
	☒ D	ROW	1L		☒ D	ROW	2L		☒ D	ROW	3L
	☒				☒				☒		
	☒				☒				☒		
	☒ GND				☒ CF(13)				☒ GND		
Pin 1	☒ CF(10)			Pin 33	☒ CF(11)			Pin 65	☒ CF(12)		

Figure 2-7. Relay Matrix Switch Module Pin-Out

The Agilent E1465A Terminal Module

Figure 2-8 shows the Agilent E1465A terminal module connectors and associated row/column designators. Refer to “Creating Larger Matrices with the Agilent E1465/66/67A Modules” later in this chapter for information on using the expansion connector.

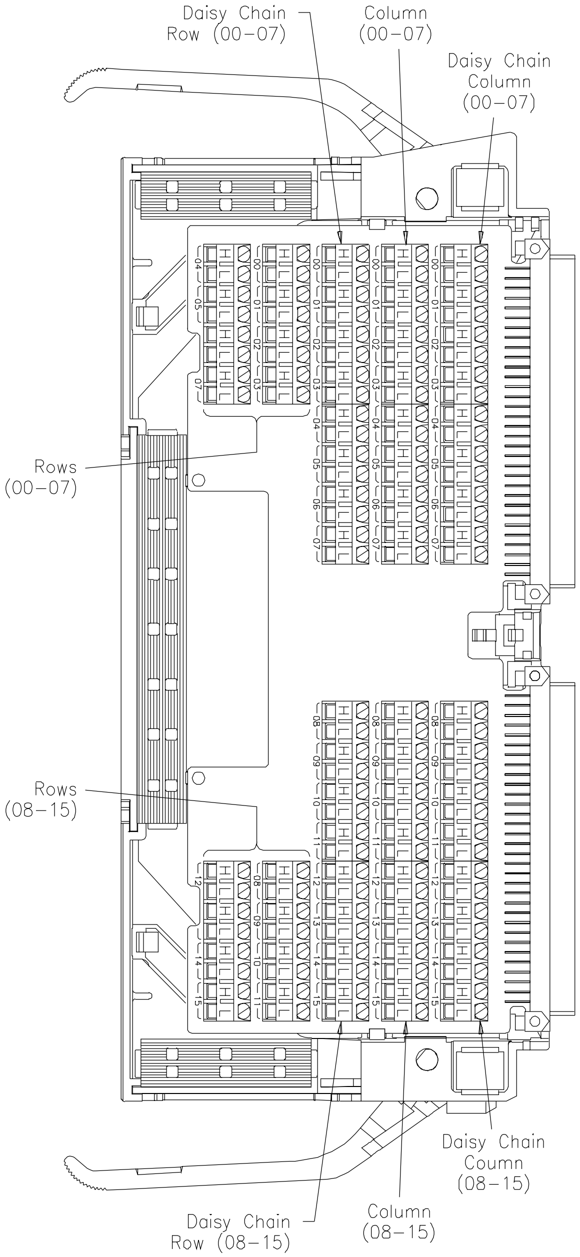


Figure 2-8. The Agilent E1465A Terminal Module

The Agilent E1466A Terminal Module

Figure 2-9 shows the Agilent E1466A terminal module connectors and associated row/column designators. Refer to “Creating Larger Matrices with the Agilent E1465/66/67A Modules” later in this chapter for information on using the expansion connector.

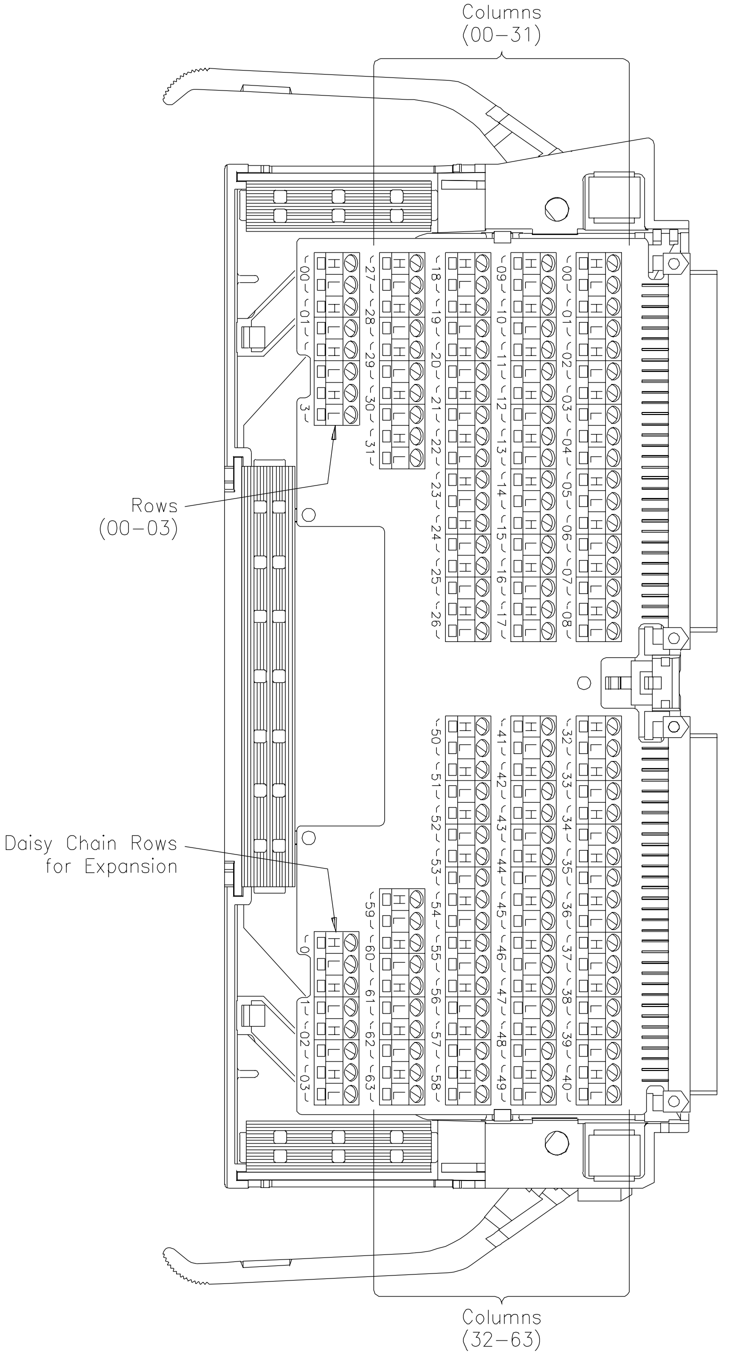


Figure 2-9. The Agilent E1466A Terminal Module

The Agilent E1467A Standard Terminal Module

Figure 2-10 shows the Agilent E1467A standard terminal module connectors and associated row/column designators. Refer to “Creating Larger Matrices with the Agilent E1465/66/67A Modules” later in this chapter for information on using the expansion connector.

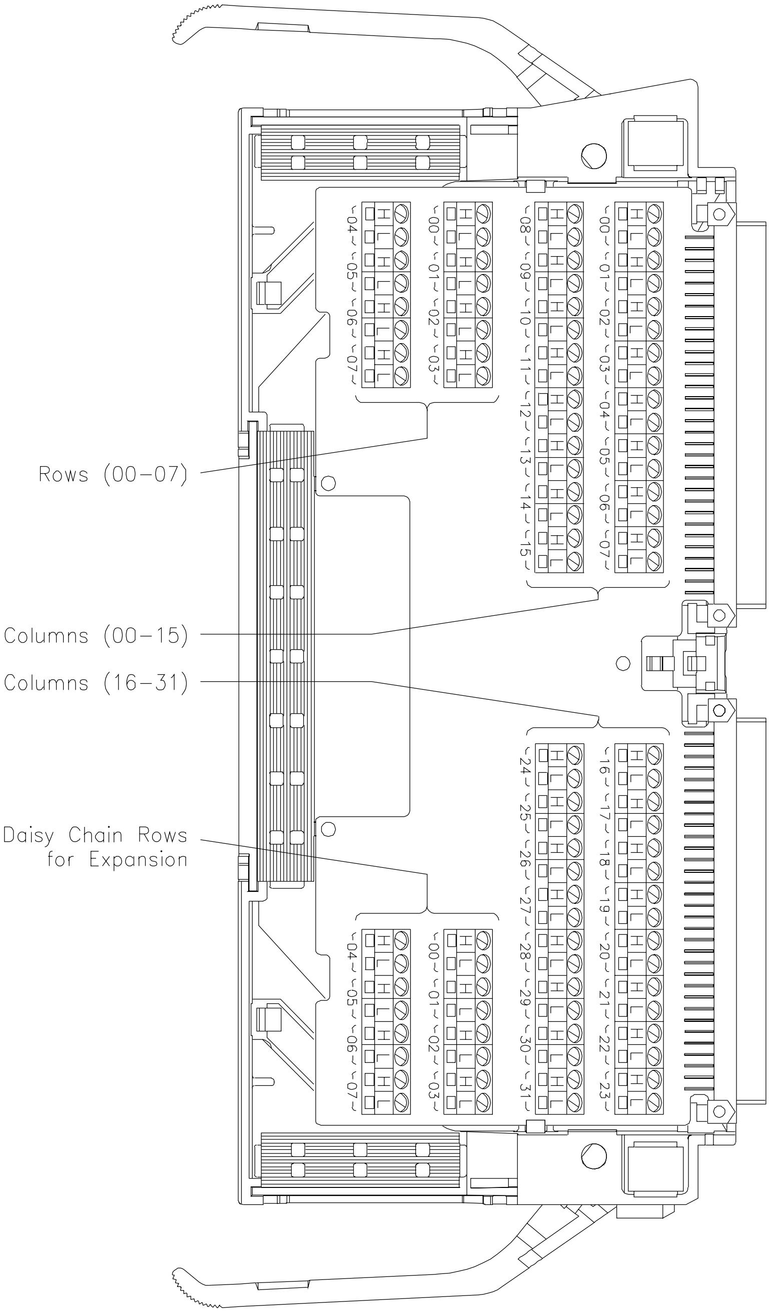


Figure 2-10. The Agilent E1467A Standard Terminal Module

For information on the Agilent E1467A Option 211 Matrix Expansion Terminal Module, refer to the section dedicated to that product later in this chapter.

Creating Larger Matrices with E1465/66/67A Modules

You can create larger matrices with the matrix modules by using the Agilent E1466-80002 Daisy Chain Expansion cable. With larger matrices more crosspoints become available. A full C-size cardcage can have up to 3,072 two-wire crosspoints. You can make a larger matrix by connecting the rows or columns of one terminal module to the corresponding rows or columns of the next terminal module. Only the Agilent E1465A has a column expansion.

Module Configuration

When using multiple modules, the modules should be configured as a switchbox. That is, the first switch card (module) has a logical address that is a multiple of 8, and succeeding switch cards have sequential logical addresses. For example, if you use the matrix default address of 120 for the first card, the remaining cards in the switchbox would have logical addresses of 121, 122, 123, and so on. See the *C-Size VXIbus Systems Configuration Guide* for additional information on switchbox configurations.

Module Address

When using multiple modules configured as a switchbox, you must address the modules as a switchbox. For example, if you want to close row 00, column 05 on the second card, you would use the following SCPI command:

```
CLOSe  (@20005)
```

Creating a 32 X 32 Matrix

Figure 2-11 shows how to connect four Agilent E1465A 16 x 16 modules to create a 32-row by 32-column matrix. This configuration requires 16 Agilent E1466-80002 Daisy Chain Expansion cables The daisy chain rows of modules 1 and 3 are connected to the rows of cards 2 and 4 (this increases the number of columns). The daisy chain columns of cards 1 and 3 are connected together, and the daisy chain columns of cards 2 and 4 are connected together. The following table summarizes which cards support which rows and columns:

Rows/Columns	**Cards (Modules)**
Rows 00 - 15	Cards 1 and 2
Rows 16 - 31	Cards 3 and 4
Columns 00 - 15	Cards 1 and 3
Columns 16 - 31	Cards 2 and 4

To connect row 16 to column 15 use the following SCPI command:

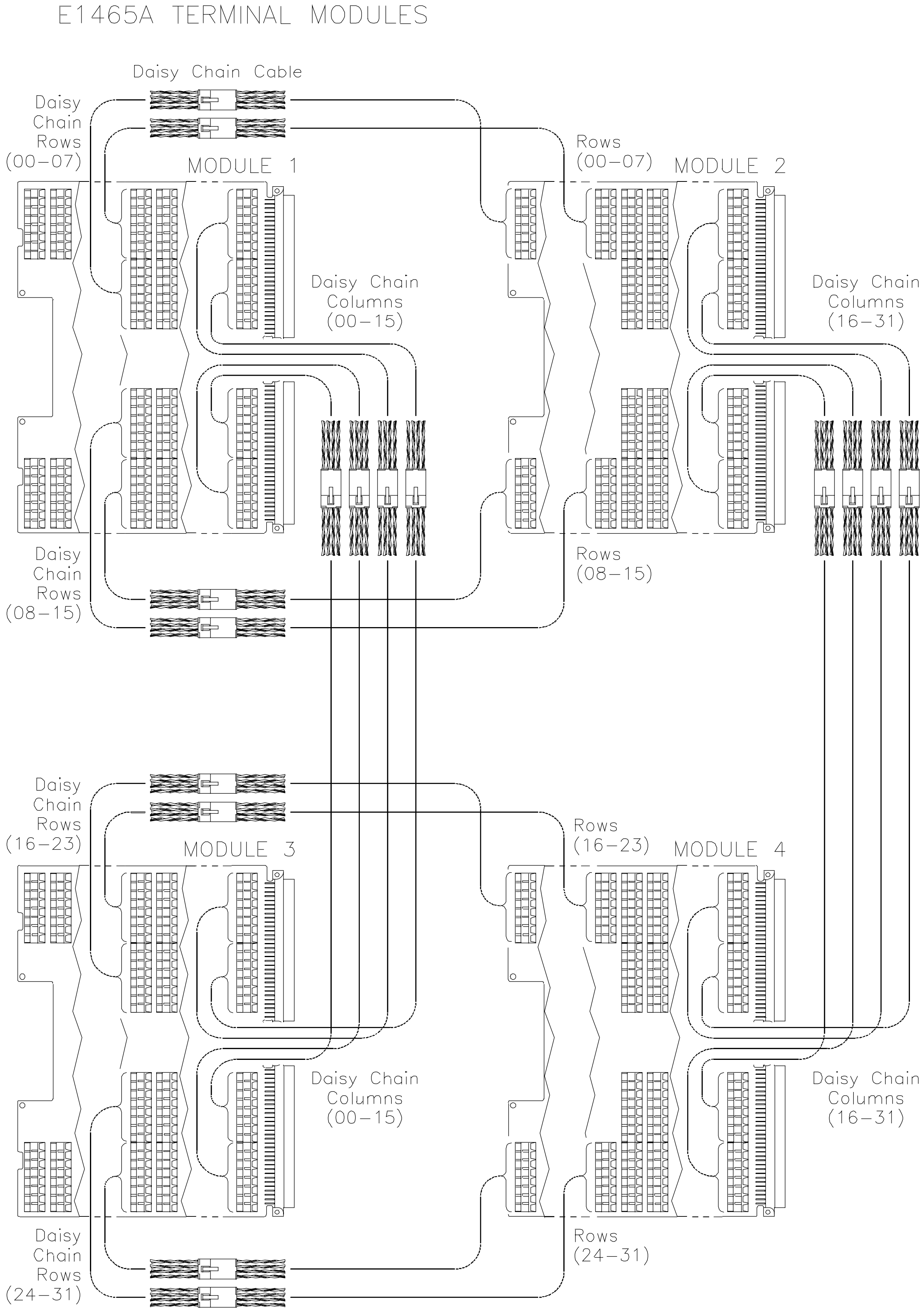


Figure 2-11. Creating a 32 X 32 Matrix

```
CLOSe (@30015)
```

See Figure 2-11. This command will close the relay on card 3, row 00, column 15.

Creating a 4 X 256 Matrix

Figure 2-12 shows how to connect four Agilent E1466A 4 x 64 modules to create a 4-row by 256-column matrix. This configuration requires three Agilent E1466-80002 Daisy Chain Expansion cables. The daisy chain rows of the first module are connected to the rows of the next module. The daisy chain rows of the second module are then connected to the rows of the next module, and so on. You can continue this pattern to create even larger matrices.

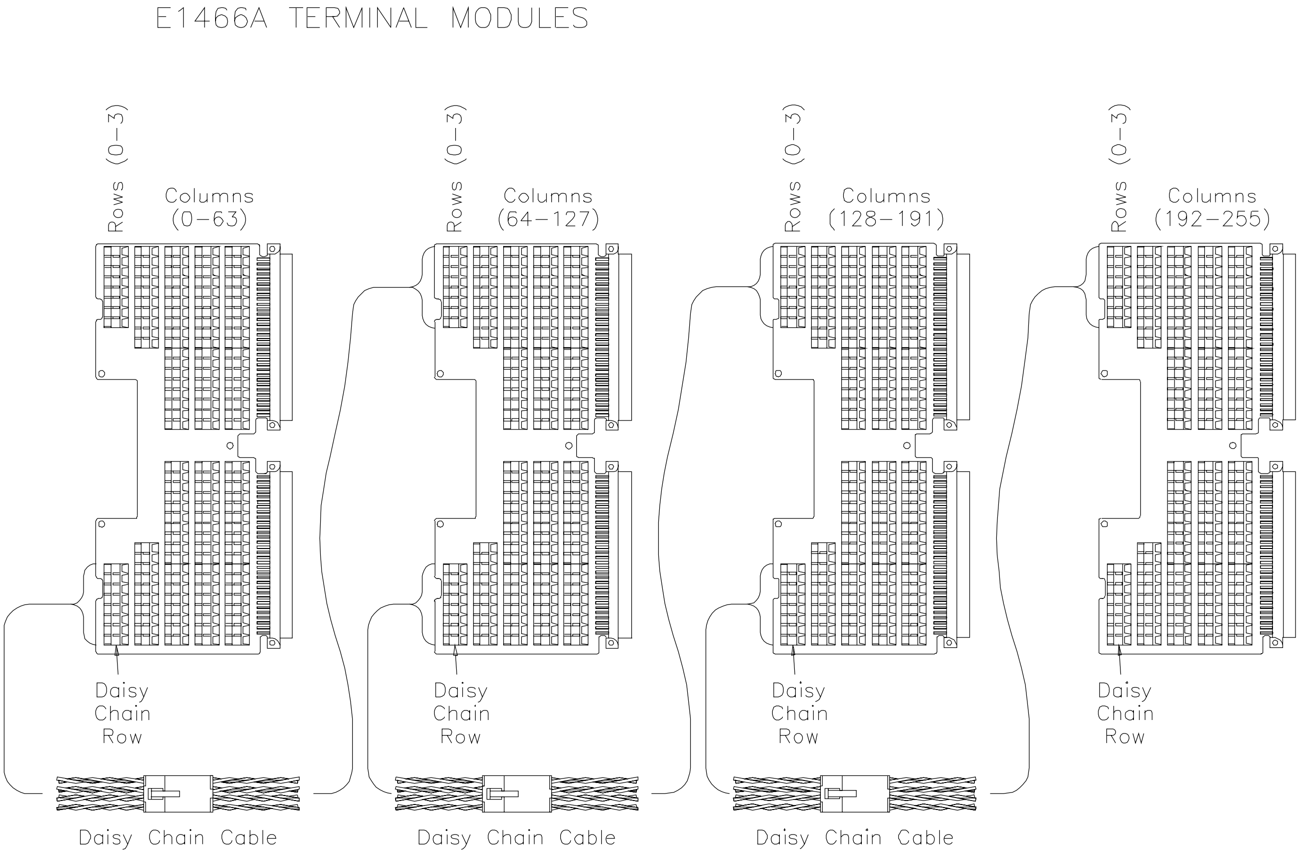


Figure 2-12. Creating a 4 X 256 Matrix

To connect row 03 to column 255, you would use the following SCPI command:

```
CLOSe (@40363)
```

See Figure 2-12. This command will close the relay on card 4, row 3, column 63.

Creating an 8 X 96 Matrix

Figure 2-13 shows how to connect three Agilent E1467A 8 x 32 modules to create an 8-row by 96-column matrix. This configuration requires four Agilent E1466-80002 Daisy Chain Expansion cables. The daisy chain rows of the first module are connected to the rows of the next module. The daisy

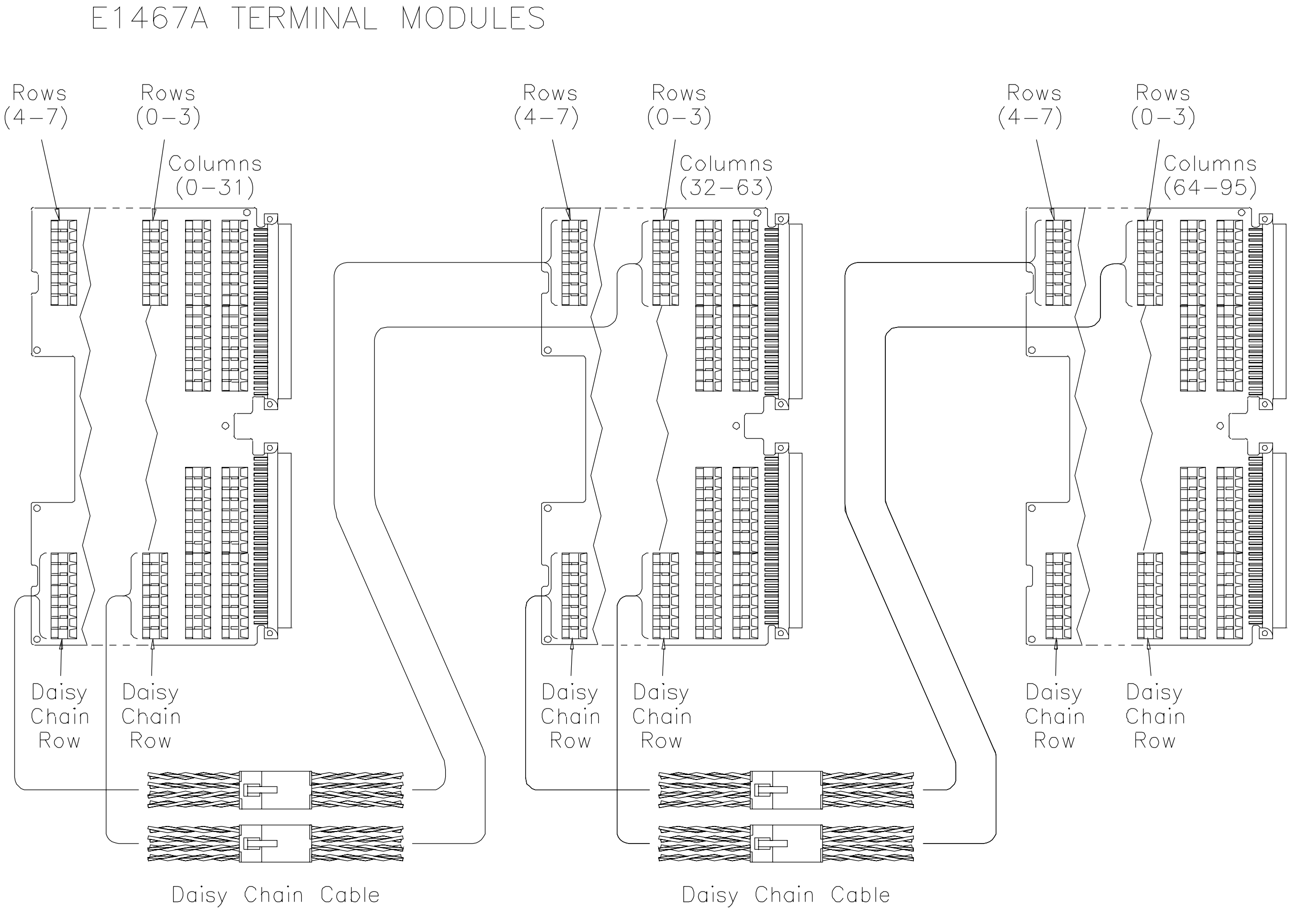


Figure 2-13. Creating an 8 X 96 Matrix

chain rows of the second module are then connected to the rows of the next module, and so on. You can continue this pattern to create even larger matrices.

To connect row 4 to column 32, execute the following SCPI command:

```
CLOSe  (@20400)
```

See Figure 2-13. This command closes the relay on card 2, row 4, column 00.

E1467A Option 211 Matrix Expansion Terminal Module

When chaining more than two Agilent E1467A 8x32 matrix modules, you may wish to use the Agilent E1467A Option 211 Matrix Expansion Terminal Module in place of the standard Agilent E1467A terminal module. Option 211 provides an 8x32 matrix configuration that can be expanded easily via the quick-access front panel cable connectors with quick connection and disconnection of expansion cables, and optimizes cable count and organization.

This module is used with the Agilent Z2220A series coax cables which are available in a variety of lengths and connections, depending on your need. Compared to the daisy-chain cable which requires each wire to be screwed into the screw terminal, the Agilent E1467A Option 211 terminal module and Agilent Z2220A series coax cables provide quicker access and easier cable connections. (Agilent will custom-quote cables to match your specific application. Contact your Agilent Sales Representative for assistance.)

Note

The Option 211 matrix expansion terminal module is specially designed for use with the Agilent E1467A and cannot be used with either the Agilent E1465A or Agilent E1466A matrix module. Option 211 replaces Option 201, which is compatible with the earlier model of the Agilent E1467A.

8-Channel Row Connectors

The Agilent E1467A Option 211 Matrix Expansion Terminal Module has one 20-pin connector which contains a high and low connection per row plus a connector shield pin. A *parallel* row connector is provided for easy chaining of multiple modules. Shield pins are common to all column connectors.

16-Channel Column Connectors

Two 34-pin connectors are provided which contain a high and low connection per column plus a column connector shield pin. A *parallel* column connector is provided for easy chaining of multiple modules. Shield pins are common to all column connectors.

Note The connector shield pins allow you to interconnect column and row shields when shielded twisted-pair cables are used. A 3-screw terminal block resides inside the terminal module, allowing connection of the column or row cable shields to each other or to chassis ground.

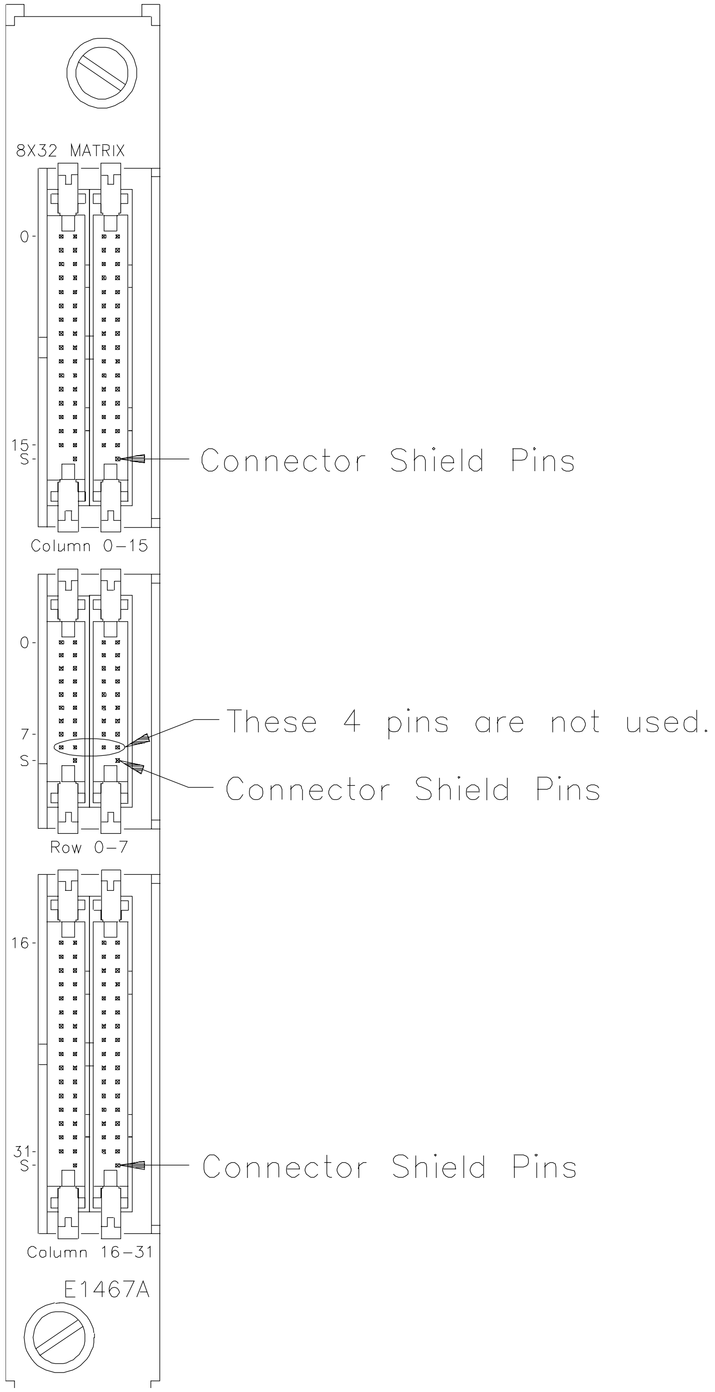


Figure 2-14. Option 211 Terminal Module Front Panel

Option 211 Expansion Terminal Module Front Panel

Figure 2-14 shows the front panel of the Agilent E1467A Option 211 Matrix Expansion Terminal Module.

Attaching the E1467A Option 211 Matrix Expansion Terminal Module

The following text and Figure 2-15 provide installation instructions for the Option 211 terminal module.

You must first install the Agilent E1467A Relay Matrix Switch Module in the mainframe, ensuring that the extraction levers are locked in place (see Figure 2-3 earlier in this chapter).

1. Using a flat-blade screwdriver, turn the expansion terminal module’s two pawl latch screws counterclockwise until the pawl is at the end of its travel. This will rotate the pawl latches out of the way of the switch module’s hood supports when joining the two modules together.
2. Align the terminal module connectors with the relay matrix switch module connectors. Engage the connectors of the two modules, using the hood supports as guides. Firmly push the expansion terminal module toward the switch module until the extraction levers on the switch module fit into the "pockets" on the side of the terminal module.
3. Turn each pawl latch screw clockwise, ensuring that the pawl rotates into position (90°) in the rectangular hole in the hood support. Once the pawl is properly positioned, turn the screw until *snug but not tight*. If the pawl does not easily rotate into the hole on the first try, make sure the two modules are engaged snugly. If this still does not allow enough clearance for the pawl to rotate into the hole, loosen the pawl latch screws (counterclockwise) until the pawl clears the hole. Then re-snug the screws.

To remove the expansion terminal module, loosen the pawl latch screws, rotating the pawls 90° so that they move out of the rectangular hole. Then firmly pull on the expansion terminal module until the connectors are no longer engaged.

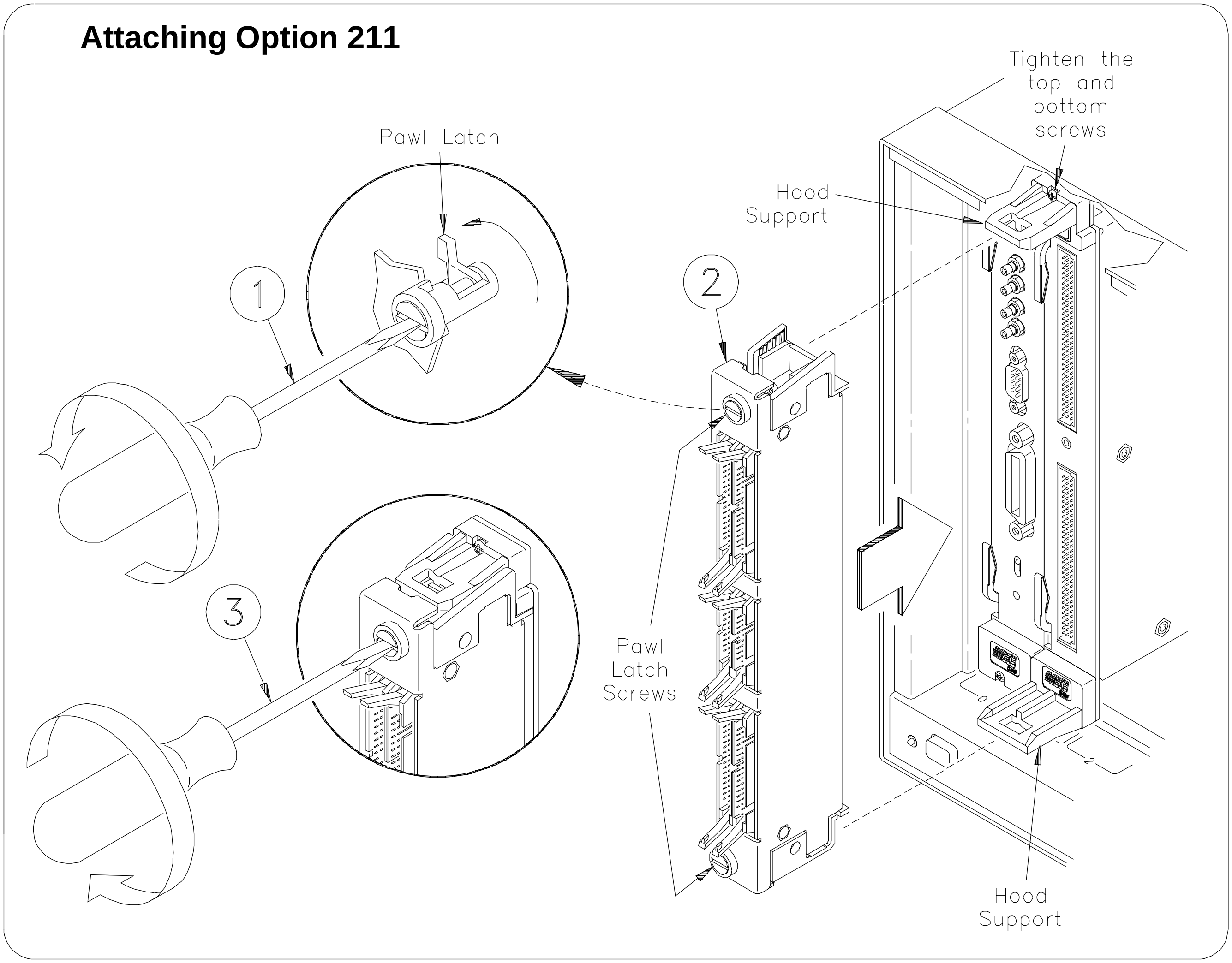


Figure 2-15. Attaching the Option 211 Terminal Module

Attaching Cables

1. Push the expansion terminal module’s ejection latches away from the center of the connector you are going to plug the cable into.
2. Inspect your cable to find the "key" in the middle of one side of the connector. (The key is a bump/raised area.) Hold the cable with the key ***on the left side*** to align with the corresponding notch in the expansion terminal module connector.
3. Firmly insert the cable into the expansion module connector, pushing until the ejection latches snap inward and grip the ends of the cable connector.

To remove the cable, pull both ejection latches away from the cable, disengaging the connectors.

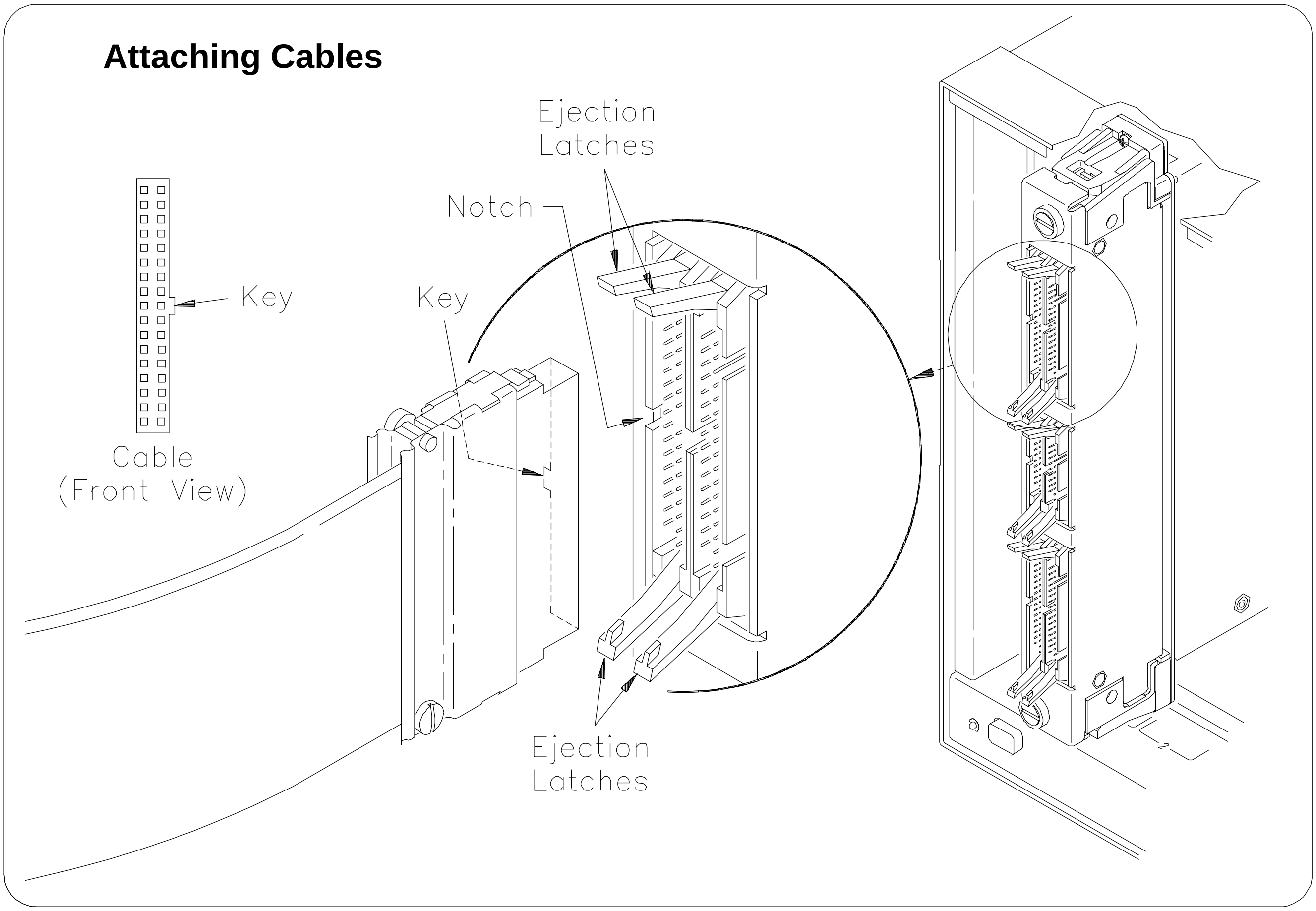


Figure 2-16. Attaching Cables to the Expansion Terminal Module

Using Multiple Mainframes

There are two ways to connect multiple mainframes together. One way is to connect the mainframes by way of GPIB and assign each mainframe a different address. See Figure 2-17. The switch modules in each mainframe are then configured as switchboxes. The switchbox card numbers are 1, 2, 3, etc. in each mainframe (each mainframe has a different address). To address the second module in the second mainframe, use the following SCPI command:

```
OUTPUT 70815; "CLOSe  (@20001)"
```

In the above command, the computer select code is 7, the command module primary address is 08, and and the matrix module’s secondary address is 15. The address (@20001) selects card 2, row 00, column 01.

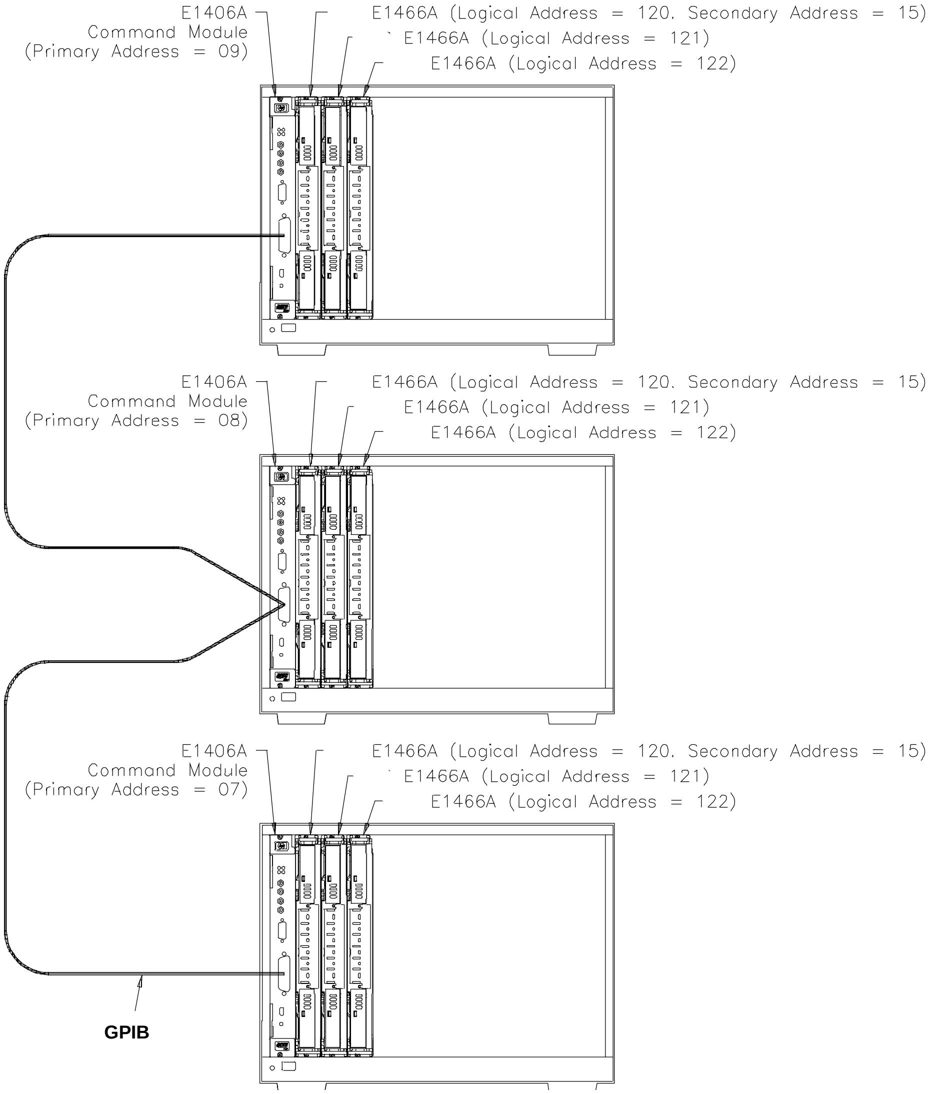

E1406A
Command Module
(Primary Address = 09)
E1466A (Logical Address = 120. Secondary Address = 15)
` E1466A (Logical Address = 121)
E1466A (Logical Address = 122)
E1406A
Command Module
(Primary Address = 08)
E1466A (Logical Address = 120. Secondary Address = 15)
E1466A (Logical Address = 121)
E1466A (Logical Address = 122)
E1406A
Command Module
(Primary Address = 07)
E1466A (Logical Address = 120. Secondary Address = 15)
` E1466A (Logical Address = 121)
E1466A (Logical Address = 122)
GPIB

Figure 2-17. Larger Matrices with Multiple Mainframes

Another way to connect multiple mainframes is to use the Agilent E1482B VXIbus Extender card. This card allows you to connect multiple mainframes via the MXIbus to look like a single mainframe. In Figure 2-18, the switch modules are again configured as a switchbox. However, the switchbox extends across multiple mainframes. The switchbox card numbers are 1, 2, 3, 4, 5, etc. extending across all the mainframes (the mainframes act as a single mainframe). To address the second card in the second mainframe, use the following SCPI command:

```
OUTPUT 70915; "CLOSe  (@40001)"
```

In the above command, the computer select code is 7, the command module primary address is 09, and the matrix module's secondary address is 15. The address (@40001) selects card 4, row 00, column 01. (See the VXI-MXI Bus Extender User's Manual for specific information on the Agilent E1482B VXIbus Expansion Module's configuration.)

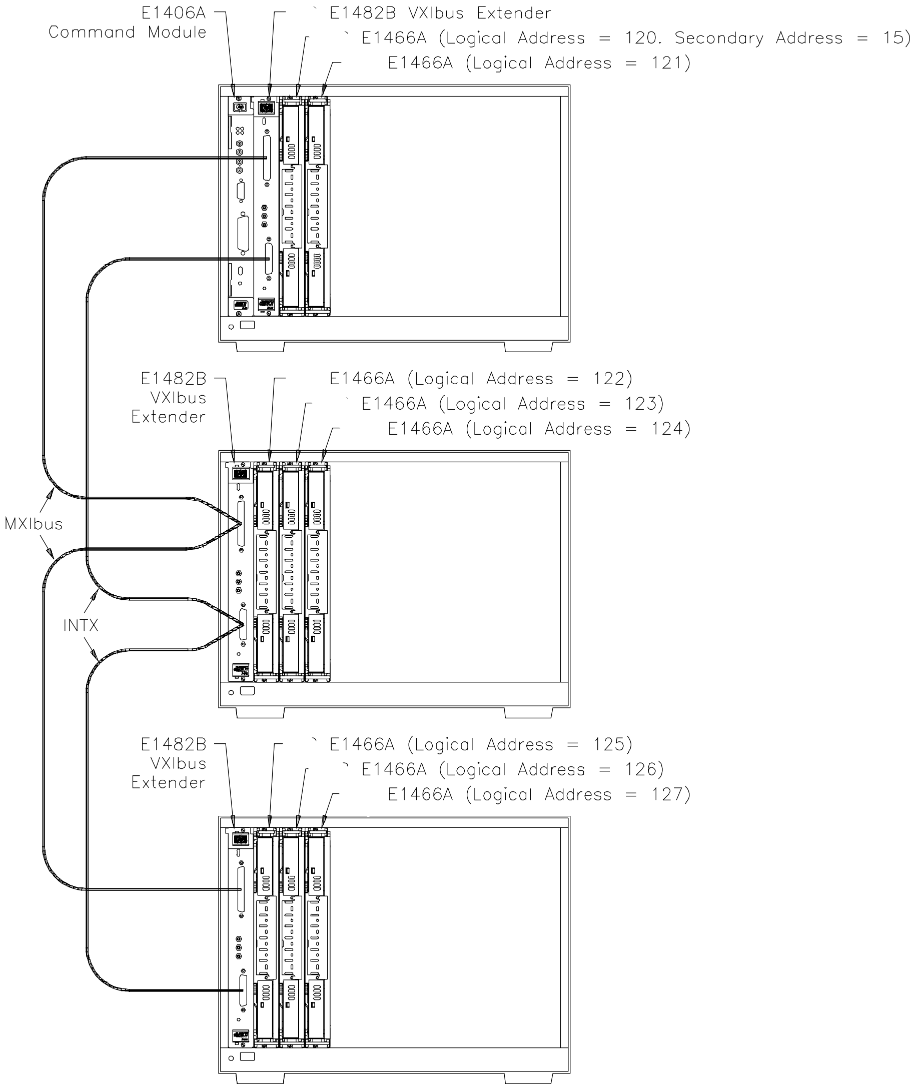

E1406A
Command Module
E1482B VXIbus Extender
E1466A (Logical Address = 120. Secondary Address = 15)
E1466A (Logical Address = 121)
E1482B
VXIbus
Extender
E1466A (Logical Address = 122)
E1466A (Logical Address = 123)
E1466A (Logical Address = 124)
MXIbus
INTX
E1482B
VXIbus
Extender
E1466A (Logical Address = 125)
E1466A (Logical Address = 126)
E1466A (Logical Address = 127)

Figure 2-18. Creating Larger Matrices Using the MXIbus

Chapter 3
Using the Matrix

Using This Chapter

This chapter uses typical examples to show how to use the matrix modules. See Chapter 4, “Relay Matrix Command Reference” for command information. Chapter contents are as follows:

All examples in this chapter use an GPIB select code of 7, primary address of 09, and secondary address of 15 (LADDR = 120) for the matrix module.

Matrix Commands

Table 3-1 explains some of the commands used in this chapter. Refer to Chapter 4 for additional information on these commands.

Table 3-1. Matrix Commands Used in Chapter 3

Command	Description
[ROUTe:]CLOSe <*channel_list*>	Closes the channels in the channel list.
[ROUTe:]CLOSe? <*channel_list*>	Queries the state of the channels in the channel list.
[ROUTe:]OPEN <*channel_list*>	Opens the channels in the channel list.
[ROUTe:]OPEN? <*channel_list*>	Queries the state of channels in the channel list.
INITiate[:IMMediate]	Starts the scan sequence and closes the first channel in the channel list.
[ROUTe:]SCAN <*channel_list*>	Defines the channel list to be scanned. Channels specified are closed one at a time.
TRIGger:SOURce <*source*>	Selects the trigger source to advance the scan.

Power-On and Reset Conditions

The matrix modules use latching relays and the relay state remains unchanged during power-up and power-down. If, however, an Agilent E1406A Command Module is used, the firmware opens all relays during power-up and a when a *RST (reset) command is executed. Table 3-2 lists the parameters and default values for the matrix module as described in this chapter.

Table 3-2. *RST (Reset) Default Conditions and Values

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

Module Identification

The following short programs use the *RST, *CLS, *IDN?, CTYP?, and CDES? commands to reset and identify the matrix modules.

BASIC

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

TURBO C

```
#include <stdio.h>
#include <chpib.h>                                  /*Include file for GPIB*/

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

main()
 {
      char into1[51], into2[51], into3[51];
       int length = 50;

                                          /*Output and enter commands to the matrix*/

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

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

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

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

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

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

A typical print for the Agilent E1465A will look like the following:

```
HEWLETT-PACKARD,SWITCHBOX,0,A.04.00
16 x 16 Matrix Switch
HEWLETT-PACKARD,E1465A,0,A.04.00
```

A typical print for the Agilent E1466A will look like the following:

```
HEWLETT-PACKARD,SWITCHBOX,0,A.04.00
4 x 64 Matrix Switch
HEWLETT-PACKARD,E1466A,0,A.04.00
```

A typical print for the Agilent E1467A will look like the following:

```
HEWLETT-PACKARD,SWITCHBOX,0,A.04.00
8 x 32 Matrix Switch
HEWLETT-PACKARD,E1467A,0,A.04.00
```

Switching Channels

Use CLOSe <*channel_list*> to close one or more matrix channels, and OPEN <*channel_list*> to open the channel(s). *channel_list* has the form @ssrrcc where ss = card number (01-99), rr is the row number, and cc = column number. Row and column definitions for the modules are as follows:

Table 3-3. Matrix Channel Numbers

Matrix Module	Rows (rr)	Columns (cc)
E1465A 16 x 16 Relay Matrix	00 - 15	00 - 15
E1466A 4 x 64 Relay Matrix	00 - 03	00 - 63
E1467A 8 x 32 Relay Matrix	00 - 07	00 - 31

Refer to Chapter 4 [ROUTe:] OPEN and [ROUTe:]CLOSe for additional information. To OPEN or CLOSe multiple channels, place a comma (**,**) between the channel numbers. For example, to close channels 10103 and 10201, execute CLOS (@10103,10201). To OPEN or CLOSe a continuous range of channels place a colon (**:**) between the first and last channel numbers.

The following BASIC program shows how to close and open row 2, column 14 on an Agilent E1465A matrix (card #1):

```
110  DISP "TEST  E1465A MATRIX"
120  OUTPUT 70915; "ROUT:CLOSE (@10214)"
130  OUTPUT 70915; "ROUT:OPEN (@10214)"
140  END
```

Note When using any of the commands that are preceded by [ROUTe:], the ROUTe portion of the command can be eliminated. For example, as in the program above, ROUTe can be eliminated and just the CLOSe command can be used.

Example: Sequencing Through the Agilent E1466A Channels

The following BASIC program sequences through each channel on an Agilent E1466A 4 x 64 Matrix Module.

```
10  OUTPUT 70915;"*RST"                 ! Reset the module.
20  FOR Row = 0 TO 3                    ! A loop to step through all the
                                        rows in the matrix.
30  FOR Col = 0 TO 63                   ! A loop to step through all the
                                        columns in the matrix.
40 Addr=10000+100*row+Col               ! Calculates channel to
                                        close.
50  OUTPUT 70915; "CLOS (@ ";Addr;")"  ! Closes the channel.
60  NEXT Col
70  NEXT Row                            ! Sequences through each row
                                        and column in the matrix.
8 0 END
```

Scanning Channels

Scanning the matrix module channels consists of closing a sequence of channels one channel at a time. Single scan, multiple scans, or continuous scanning modes are available.

The TRIGger:SOURce command specifies the source to advance the scan. The OUTPut command can be used to enable the Agilent E1406A Command Module Trig Out port or TTL Trigger bus lines (0-7).

Example: Scanning Channels with System Multimeter Using TTL Trigger

This example uses the command module's TTL Trigger Bus Lines to synchronize matrix channel closures to a system multimeter (Agilent E1412A). For measurement synchronization:

- Agilent E1406A TTL Trigger Bus Line 0 is used by the matrix module to trigger the multimeter to perform a measurement.
- Agilent E1406A TTL Trigger Bus Line 1 is used by the multimeter to advance the matrix scan.

These trigger bus lines are not actual hardware connections. The triggering is accomplished by the Agilent E1406A's firmware. Row 00 (High and Low) of an Agilent E1465A 16 x 16 Relay Matrix Module is connected to the voltmeter's High and Low. The columns are then scanned, switching in different DUTS (devices under test). Figure 3-1 shows how to connect the matrix module to the multimeter module. The connections shown with dotted lines are not actual hardware connections. These connections indicate how the firmware operates to accomplish the triggering.

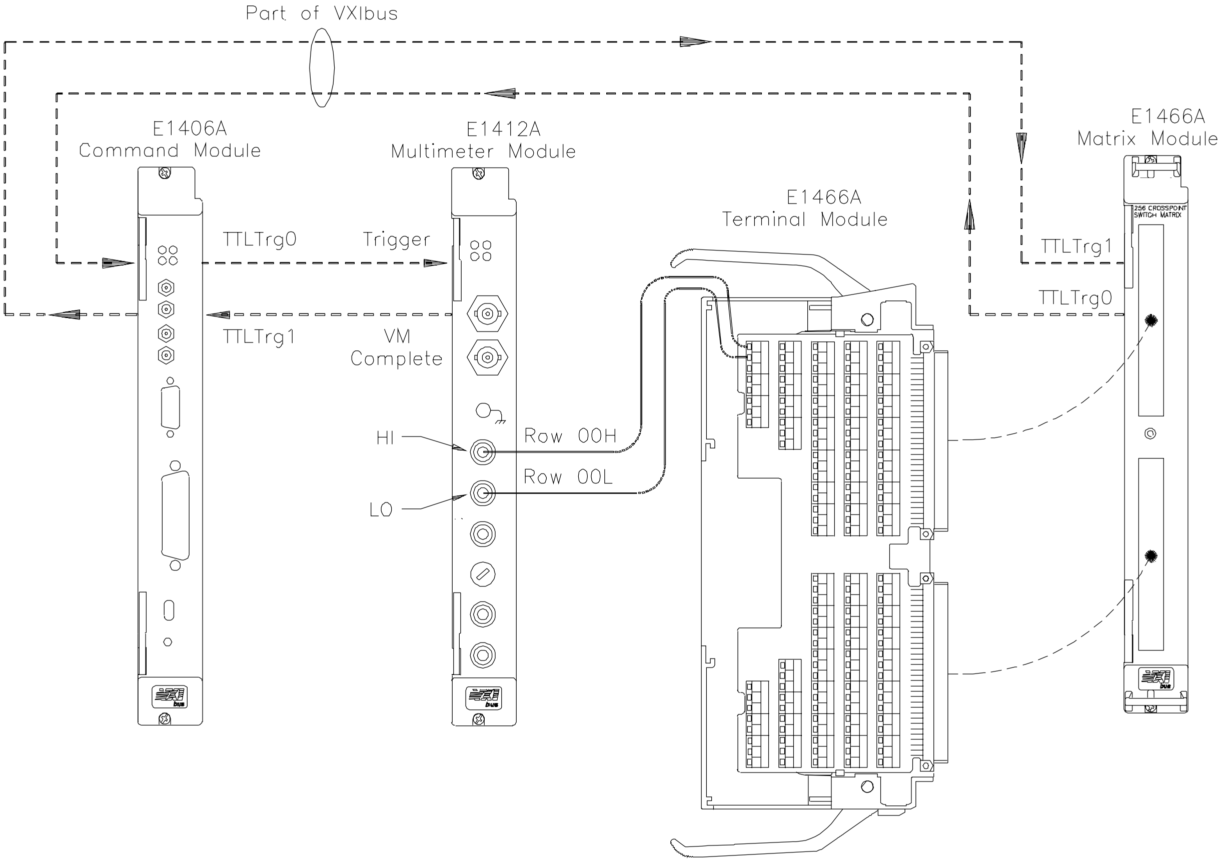


Figure 3-1. Example: TTL Trigger Bus Scanning

The following BASIC program sets up the voltmeter (GPIB address 70903) to scan making two-wire resistance measurements. The matrix is set to scan row 00, columns 00 to 15 on an Agilent E1465A Matrix Module:

```
10 ALLOCATE REAL Rdgs(1:16)
       ! Reset and clear the modules.
20 OUTPUT 70915; "*RST;*CLS"
30 OUTPUT 70903; "*RST;*CLS"
       ! VM Triggers on TTL Trigger line 0, VM pulses TTL Trigger line 1 on
       ! measurement complete.  Set VM function to Resistance, range, NPLC.
40 OUTPUT 70903; "ABORT;:TRIG:SOUR TTLTRG0"
50 OUTPUT 70903; "OUTP:TTLTRG1:STAT ON"
60 OUTPUT 70903; "CONF:RES AUTO,DEF"
70 OUTPUT 70903; "TRIG:DEL 0; COUN 16;:CAL:ZERO:AUTO ON"
       ! Check to see if VM ready; when ready, initialize trigger.
80 OUTPUT 70903; "*OPC?"
90 ENTER 70903; Check
100 OUTPUT 70903; "INIT"
      ! Set matrix to be triggered by TTL Trigger line 1.
 110 OUTPUT 70915; "TRIG:SOUR TTLTRG1"
       ! Set up the matrix module: matrix pulses TTL Trigger line 0 on
       ! channel closed, scan list is Row 0, Columns 0 to 15.  Initiate scan.
120 OUTPUT 70915; "OUTPUT:TTLT0:STATE ON"
130 OUTPUT 70915; "SCAN (@10000:10015)"
140 OUTPUT 70915; "INIT"
       ! Enter and print readings.
150 OUTPUT 70903; "FETCH?"
160 ENTER 70903; Rdgs(*)
170 PRINT Rdgs(*)
180 END
```

Example: Scanning Channels with an External Instrument Using Trigger In and Out

The following example uses the Agilent E1406A Command Module Trig In and Trig Out ports to synchronize the matrix module channel closures to an external Agilent 3457A voltmeter. Figure 3-2 shows how to connect the voltmeter to the command and matrix modules. Note that the command module’s Trig In port connects to the voltmeter’s Complete port, and the command module’s Trig Out port connects to the voltmeter’s External Trigger port. The external voltmeter is at address 722.

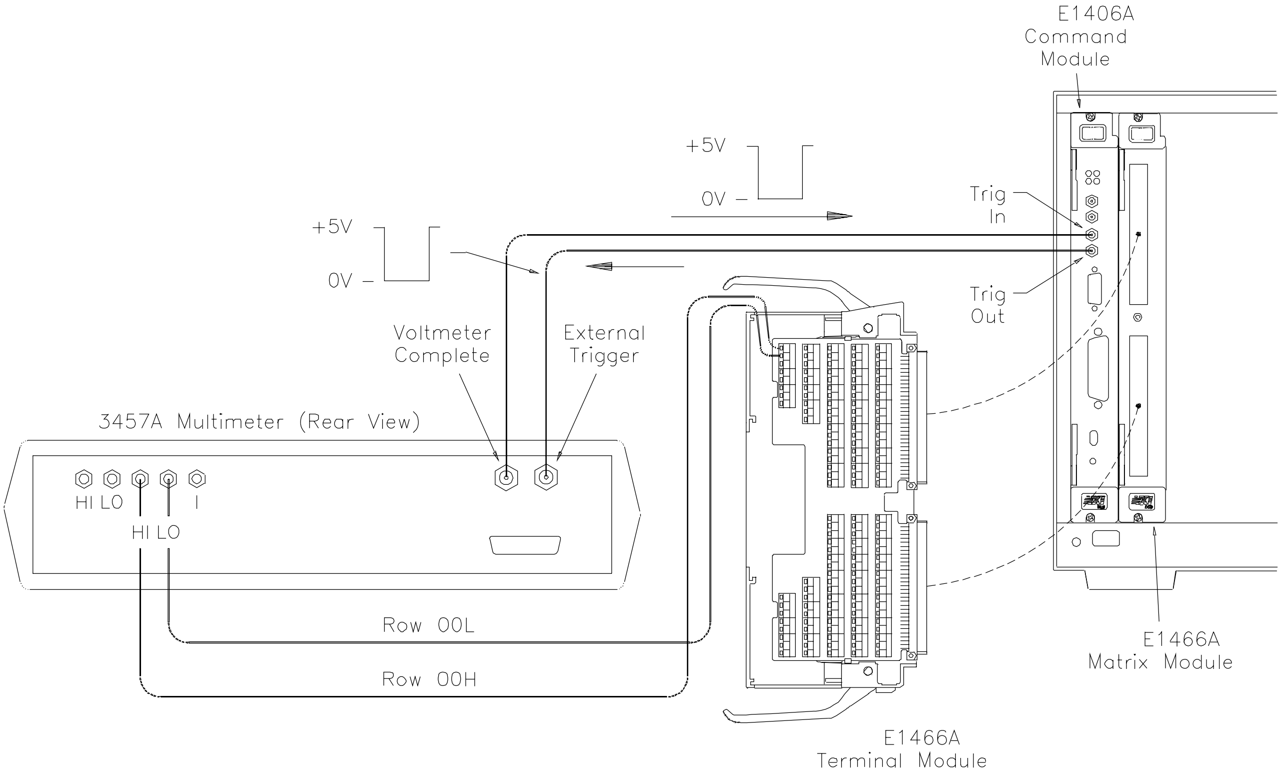


Figure 3-2. Scanning with an External Voltmeter

```
        ! Set up voltmeter for external trigger, DCV measurements, memory
        ! first in first out storage.
10 OUTPUT 722; "TRIG EXT; DCV;MEM FIFO"
        ! Reset the matrix, enable the Agilent E1406A Trig Out port, set trigger
        ! source to external triggering.
20 OUTPUT 70915; "*RST;*CLS"
30 OUTPUT 70915; "OUTP ON"
40 OUTPUT 70915; "TRIG:SOUR:EXT"
        ! Set matrix measurement mode, define channel list, initiate scan.
50 OUTPUT 70915; "SCAN (@10000:10015)"
60 OUTPUT 70915; "INIT"
70 WAIT 2
80 FOR Channels = 1 to 16
90 ENTER 722;Results
100 PRINT Results
110 NEXT Channels
120 END
```

Querying the Matrix Module

All query commands end with a "?". These commands are used to determine a specific state of the module. The data is sent to the output buffer where you can retrieve it into your computer. See Chapter 4 for more information on these commands.

The CLOSe? <*channel_list*> and OPEN? <*channel_list*> commands return the current state of the specified channel. These commands return "1" if the operation is true, and "0" if false. See Chapter 4, [ROUTe:]OPEN? and [ROUTe:] CLOSe? commands for additional information on these commands.

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

The following example closes a range of channels on the Agilent E1467A 8 x 32 Matrix Module and queries for the results:

```
      ! Dimensions two string variables to 128 characters.
10  DIM Chan1$[128], Chan2$[128]
      !Closes rows 00 through 07, columns 00 through 31.
20  OUTPUT 70915;"CLOS (@10000:10731)"
      ! Queries rows 00 through 03, columns 00 through 31 to see if the
      ! channels are closed.
30  OUTPUT 70915; "CLOS? (@10000:10331)"
      ! Enter the results of the first 128 channel closures.
40  ENTER 70915; Chan1$
      ! Queries rows 04 through 07, columns 00 through 31 to see if the
      ! channels are closed.
50  OUTPUT 70914; "CLOS? (@10400:10731)"
      ! Enter the results of the second 128 channel closures.
60  ENTER 70915; Chan2$
       ! Prints all the channels that are closed (should print 1's).
70  PRINT "Channels closed";Chan1$, Chan2$!
80  END
```

Using the Scan Complete Bit

You can use the Scan Complete Bit (bit 8) in the Operation Status Register (command module) 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 using the STAT:OPER? command. Refer to the STATus:OPERation[:EVENt]? command in Chapter 4.

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 128 Common Command to assert an GPIB Service Request (SRQ), you can interrupt the computer when the Scan Complete Bit is set, after the scanning cycle completes. This allows the controller to do other operations while the scanning cycle is in progress.

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

```
        ! Reset and Clear the Matrix.
10 OUTPUT 70915;"*RST; *CLS"
         ! Enable Scan Complete Bit.
20 OUTPUT 70915; "STATUS:OPER:ENABLE 256"
        ! Set the matrix up for continuous triggering.
30 OUTPUT 70915; "TRIG:SOUR IMM"
        ! Select channels to scan.
40 OUTPUT 70915; "SCAN (@10000:10015)"
        ! Wait for operation complete.
50 OUTPUT 70915; "*OPC?"
60 ENTER 70915; A$
70 PRINT "*OPC? = ";A$
        ! Query the contents in the operation status register.
80 OUTPUT 70915; "STAT:OPER:ENAB?"
90 ENTER 70915; A$
100 PRINT "STAT:OPER:ENAB? = ";A$
```

Continued on next page.

```
        ! Query the contents of the status byte register.
110  OUTPUT 70915; "*STB?"
120  ENTER 70915; A$
130  PRINT "Switch Status = ";A$
        ! Start scan cycle.
140  OUTPUT 70915; "INIT"
        ! Initialize the value of the counter.
150  I = 0
        ! Stay in loop until some value is returned from the SPOLL (70915)
        ! command.
160  WHILE (I=0)
170     I = SPOLL(70915)
180     PRINT "Waiting for scan to complete: SPOLL = ";I
190  END WHILE
200  I = SPOLL(70915)
210  PRINT "Scan complete: spoll = ";I
220  END
```

Recalling and Saving States

This section contains information about saving and recalling current matrix module states.

Storing States

The *SAV <*numeric_state*> command stores the current state of the matrix channels. Up to 10 states can be stored by specifying the <*numeric_state*> as an integer 0 through 9. The following states are stored:

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

Recalling States

The *RCL <*numeric_state*> command recalls the specified previously stored state. If the specified <*numeric_state*> does not exist, the matrix module configures to its power-on/reset states (refer to Table 3-2). The following program shows how to save and recall matrix switch states:

```
        ! Dimensions a string variable A$ to 30 characters.
10 DIM A$[30]
        ! Close channels on the matrix module.
20 OUTPUT 70915; "CLOS (@10000:10015)
```

Continued on next page.

```
            ! Save as numeric state 5.
30 OUTPUT 70915; "*SAV 5"
       ! Reset and clear the module.
40 OUTPUT 70915; "*RST; *CLS"
       ! Query to see what channels are closed.
50 OUTPUT 70915; "CLOS? (@10000:10020)"
60 ENTER 70915;A$
70 PRINT "Channels Closed:";A$
       ! Recall numeric state 5.
80 OUTPUT 70915; "*RCL 5"
        ! Check to see if the recalled channels are closed.
90 OUTPUT 70915; "CLOS? (@10000:10200)"
100 ENTER 70915; A$
       ! Prints 1s for the first 16 channels that are closed and 0s for the
      ! remaining 5 channels.
110 PRINT "Channels Closed:";A$
120 END
```

Detecting Error Conditions

The SYSTem:ERRor? query requests a value from instrument's error register. This register contains an integer in the range [-32768 to 32767]. The response takes the following form:

<err_number>,<err_message>

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

The following programs attempt an illegal channel closure for the Agilent E1466A (4 X 64 Matrix) and poll for the error message:

BASIC

```
10   DIM Err_num$[256]                  ! Dimensions a string variable,
                                        Err_num$ for 256 characters.
20   OUTPUT 70915; "CLOS (@10500)"      ! Try to close an illegal channel.
30   OUTPUT 70915; "SYST:ERR?"          ! Check  for a system error.
40   ENTER 70915; Err_num$              ! Enter the errors into
                                        Err_num$.
50   PRINT Err_num$                     ! Prints error +2001, "Invalid
                                        channel number".
60    END
```

TURBO C

```
#include <stdio.h>
#include <chpib.h>                               /*Include file for GPIB*/

#define ISC 7L
#define MATRIX 70915L                                /*Matrix default address*/
#define TASK1 "CLOSE (@10500)"                   /*Command for illegal switch closure*/
#define TASK2 "SYST:ERR?"                        /*Command for system error*/

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

                                                 /*Output commands to matrix*/

     error_handler (IOTIMEOUT (7L,5.0), "TIMEOUT");
     error_handler (IOOUTPUTS (MATRIX, TASK1, 15), "OUTPUT command");
     error_handler (IOOUTPUTS (MATRIX, TASK2, 9), "OUTPUT command");

                                                 /*Enter from matrix*/

     error_handler (IOENTERS (MATRIX, into, &length), "ENTER command");
     printf("Now let’s print the errors: %s",into);

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

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

Synchronizing the Matrix Module

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

```
10 OUTPUT 70915; "*RST"                   ! Reset the module.
20 OUTPUT 70915; "CLOS (@10012)"          ! Close a channel.
30 OUTPUT 70915; "*OPC?"                  ! Wait for operation complete.
40 ENTER 70915; Opc_value
50 OUTPUT 70915; "CLOS? (@10012)"         ! Test that the channel is closed.
60 ENTER 70915;A
70 IF A=1 THEN
80        OUTPUT 70903;"MEAS:VOLT:DC?" ! When channel is closed,
                                          measure the voltage.
90        ENTER 70903; Meas_value         ! Print the measured value.
100       PRINT Meas_value
110 ELSE
          PRINT "CHANNEL DID NOT CLOSE"
120 END IF
130 END
```

Understanding the Relay Matrix

This section provides additional details about the Agilent E1465/66/67A relay matrices. It explains the internal configuration so you can better understand how each matrix operates.

Advantages of Latching Relays

There are several advantages to using the Agilent E1465A/E1466A/E1467A latching relays. First, with 256 relays on the dense matrix relay module, latching relays prevent excessive current drawn from the power supply if the user closes too many relays accidentally. Second, energy is saved since power is not continually applied to keep a latching relay closed. Third, by not continually applying power, the relay coil does not heat up. This is important because the two metal contacts inside the relay, in effect, form a thermocouple. Thus, temperature differences on the relay contacts cause thermal EMF (electromotive force) to be generated. Also, the life of the latching relay is usually longer than the nonlatching relay because of the power that must be continually applied to close a nonlatching relay.

The main disadvantage of latching relays is that the relay state is unchanged at power-on, power-off, or following a reset. Therefore, the device's firmware must ensure that all relays are open following these conditions.

Increased System Throughput

In conventional switch module designs, the module interrupts the central processing unit (CPU) each time a relay is opened or closed. In the design of the Agilent E1465A/E1466A/E1467A matrix relay modules, the CPU is interrupted one time after all relays in the specified channel list have been opened or closed. Thus, system throughput speed is increased.

The following describes the relay module functions (see Figure 3-3):

- A command is sent to the relay module and stored in the FIFO memory.
- Once the data is in memory, the VME Timing PAL (programmable array logic) asserts DTACK*. This signals the CPU on the relay module's commander that it is now free to service other tasks.
- The VME Timing PAL signals the FIFO Interface PAL to execute the command. During execution, the Data Bus FIFO EMPTY* flag signals the FIFO Interface PAL to read the Data Bus and Address Bus FIFO and generate 7-ms pulses to activate the relays. Only one 7-ms pulse is required per relay bank (up to 16 relays).
- The FIFO Interface PAL reads the Data Bus and Address Bus FIFO until the EMPTY* flag signals the FIFO Interface PAL the FIFO memory is empty.

- When the FIFO is empty, the FIFO Interface PAL signals the VME Timing PAL which asserts IRQ*. This interrupts the command module CPU after the last relay has been activated.

Because the relay module asserts IRQ* after the last relay is activated, the CPU is not continually interrupted; thus, system throughput is enhanced.

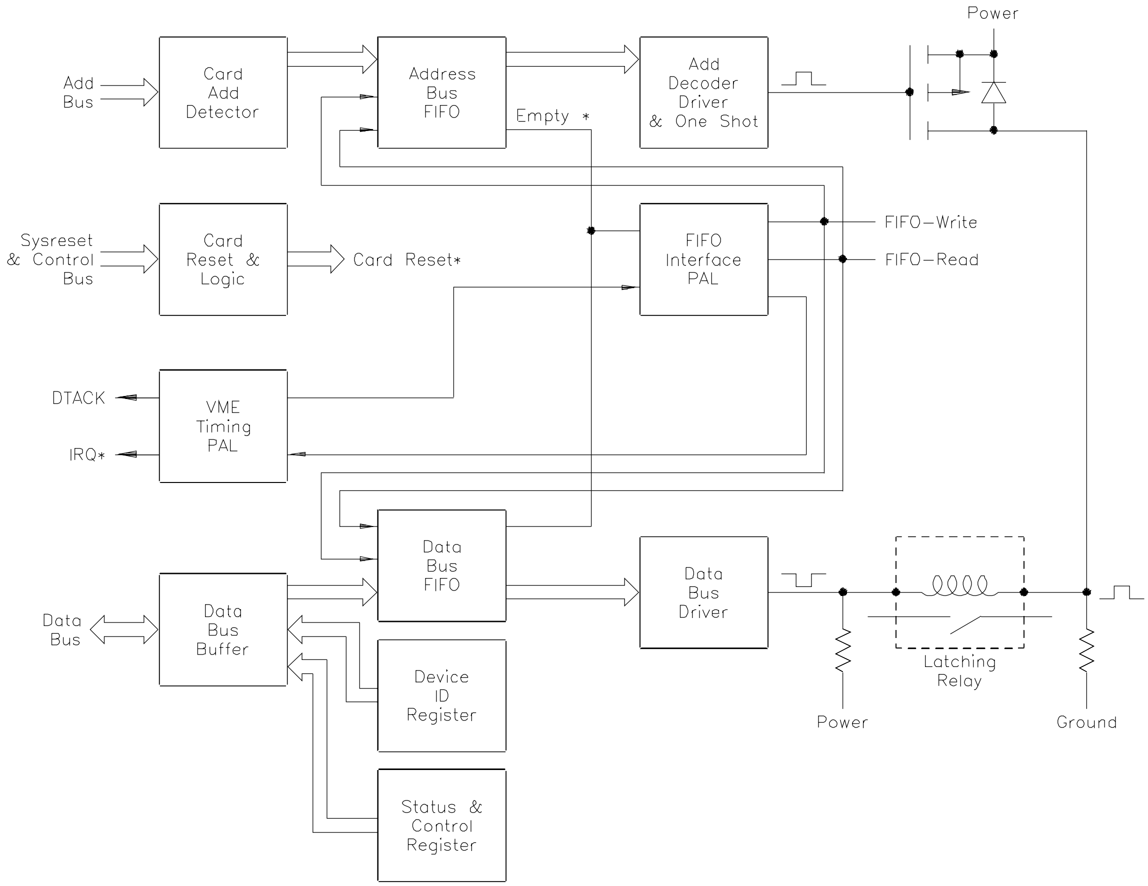


Figure 3-3. Matrix Module Block Diagram

Chapter 4
Relay Matrix Command Reference

Using This Chapter

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

See the *Agilent E1406 Command Module User's Manual* for additional information on SCPI and common commands. Chapter contents are as follows:

Command Types

Commands are separated into two types: IEEE 488.2 Common commands and SCPI commands. Each of these types of commands are discussed in the following sections.

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>
```

ROUTe: is the root command and CLOSe and SCAN are second-level commands.

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

Command Separator

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

```
[ROUTe:]SCAN
```

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

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, and for better program readability, send the entire command. The instrument will accept either the abbreviated form or the entire command.

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

Implied Commands

Implied commands are those which appear in square brackets ([ ]) in the command syntax. (Note that the brackets are not part of the command and are not sent to the instrument.) You can send a second-level command but you do not need to 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 which follows:

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

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

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

Parameters

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

Parameter Type	Explanations and Examples
Numeric	Accepts all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation. 123, 1.23E2, –123, –1.23E2, .123, 1.23E–2, 1.23000E–01. Special cases include MINimum, MAXimum, and INFinity.
Boolean	Represents a single binary condition that is either true or false. ON, OFF, 1, 0
Discrete	Selects from a finite set of values. These parameters use mnemonics to represent each valid setting. An example is TRIGger:SOURce <*source*> 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 current number of scanning cycles is returned. If you send the MIN parameter, the command returns the minimum available (1). If you send the MAX parameter, the command returns the maximum available (32,767). 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;*RST
```

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 relay matrix. Commands are listed alphabetically by subsystem and within each subsystem.

ABORt

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

Subsystem Syntax

ABORt

Comments

- **ABORt Actions:** ABORt aborts the scan and invalidates the current channel list.

- **Stopping Scan Enabled via Interface:** When a scan is enabled by way of an interface, an interface clear command (CLEAR 7) can be used to stop the scan. When the scan is enabled via the interface and TRIG:SOUR BUS or HOLD is set, you can use ABORt to stop the scan.

- **Restarting a Scan:** Use the INIT command to restart the scan.

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

Example

Stopping a Scan with ABORt

This example stops a (continuous) scan in progress.

```
TRIG:SOUR BUS                 *TRG command is trigger
                              source.
INIT:CONT ON                  Set continuous scanning.
SCAN (@10000:10003)           Scan channels 00-03.
INIT                          Start scan, close channel 00.
   .
   .
ABOR                          Abort scan in progress.
```

ARM

The ARM subsystem selects the number of scanning cycles (1 to 32767) for each INITiate command.

Subsystem Syntax

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

:COUNt

ARM:COUNt <*number*> allows scanning cycles to occur a multiple of times (1 to 32767) with one INITiate command when INITiate:CONTinuous OFF | 0 is set. MIN sets 1 cycle and MAX sets 32,767 cycles.

Parameters

Parameter Name	Parameter Type	Range of Values
<*number*>	numeric	1 – 32767 | MIN | MAX

Comments

- **Number of Scans:** Use only numeric values between 1 and 32767 for the number of scanning cycles.

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

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

Example

Setting Ten Scanning Cycles

This example sets a relay matrix for 10 scans of channels 10000 through 10003. When the scan sequence completes, channels 10000 through 10003 are closed.

```
ARM:COUN 10                  10 scans per INIT command.
SCAN  (@10000:10003)         Scan channels 10000-10003.
INIT                         Start scan, close channel
                             10000.
```

:COUNt?

ARM:COUNt? [MIN | MAX] returns the current number of scanning cycles set by ARM:COUNt. The current number of scan cycles is returned when MIN or MAX is not specified. With MIN or MAX as a parameter, MIN returns 1 and MAX returns 32767.

Parameters

Parameter Name	Parameter Type	Range of Values
MIN | MAX	numeric	MIN = 1, MAX =32767

Comments

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

Example

Query Number of Scans

This example sets a switchbox for 10 scanning cycles of channels 10000 through 10003 and queries the number of scan cycles set. The ARM:COUN? command returns 10.

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

DISPlay

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

Subsystem Syntax

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

:MONitor[:STATe]

DISPlay:MONitor[:STATe] <*boolean*> turns the monitor mode on or off.

Parameters

Parameter Name	Parameter Type	Range of Values
STATe	boolean	ON | OFF | 1 | 0

Comments

- **Monitoring Switchbox Channels:** DISPlay:MONitor[STATe] ON or DISPlay:MONitor[:STATe] 1 turns the monitor mode *on* to show the channel state of the selected module. DISPlay:MONitor[:STATe] OFF or DISPlay:MONitor[:STATe] 0 turns the channel monitor *off*.

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

- **Monitor Mode with an Agilent E1466A/67A/68A:** When monitor mode is turn on, a hexadecimal number representing the channels closed will be displayed at the bottom of the terminal. For example, an Agilent E1466A with row 0, columns 0-3 closed will look like to following:

```
R0: 0000 0000 0000 000F  R1: 0000 0000 0000 0000   R2:  0000 0000  . . . etc.
```

Where 000F$_h$ represents columns 0-15. F (1111) represents channels closures for columns 0-3. If 000C$_h$ was listed instead, then columns 2 and 3 (1100) would be closed.

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

Example

Enabling the Monitor Mode

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

:MONitor:CARD

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

Parameters

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

Comments

- **Selecting a Specific Module to be Monitored:** Use the DISPlay:MONitor:CARD command to send the card number for the switchbox to be monitored.

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

- ***RST Conditions:** DISPlay:MONitor:CARD AUTO

Example

Select Module #2 in a Switchbox for Monitoring

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

INITiate

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

Subsystem Syntax

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

:CONTinuous

INITiate:CONtinuous <*mode*> enables or disables continuous scanning cycles for the matrix.

Parameters

Parameter Name	Parameter Type	Range of Values
<*mode*>	boolean	ON | OFF | 1 | 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 specified by the TRIGger:SOURce command advances the scan through the channel list. A trigger at the end of the channel list closes the first channel in the channel list and the scan cycle repeats.

- **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. At the end of the scanning cycle, the last channel in the channel list is opened.

- **Stopping Continuous Scan:** See the ABORt command.

- **Related Commands:** ABORt, ARM:COUNt, TRIGger, TRIGger:SOURce

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

Example

Enabling Continuous Scanning

This example enables continuous scanning of channels 10000 through 10003 of a single module. Since TRIGger:SOURce IMMediate (default) is set, the example uses an interface clear command (such as CLEAR) to stop the scan.

INIT:CONT ON	*Enable continuous scanning.*
SCAN (@10000:10003)	*Defines channel list.*
INIT	*Start scan cycle, close channel 10000.*

:CONTinuous?

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

Example

Query Continuous Scanning State

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

INIT:CONT ON	*Enable continuous scanning.*
INIT:CONT?	*Query continuous scanning state.*

[:IMMediate]

INITiate[:IMMediate] starts the scanning process and closes the first channel in the channel list. Successive triggers from the source selected 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. Each trigger received advances the scan to the next channel in the channel list. An invalid channel list definition causes an error (see [ROUTe:]SCAN).

- **Stopping Scanning Cycles:** See the ABORt command.

Example

Enabling a Single Scan

This example enables a single scan of channels 10000 through 10003 of a matrix module. The trigger source to advance the scan is immediate (internal) triggering set with (default) TRIGger:SOURce IMMediate.

SCAN (@10000:10003)	*Scan channels 10000-10003.*
INIT	*Begin scan, close channels 10000 (use immediate triggering).*

OUTPut

The OUTPut command subsystem enables or disables the different trigger lines of the Agilent E1406A Command Module.

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

:EXTernal[:STATe]

OUTPut:EXTernal[:STATe] <*mode*> enables or disables the Trig Out port on the Agilent E1406A Command Module to output a trigger when a channel is closed during a scan. ON | 1 enables the port and OFF | 0 disables the port.

Parameters

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

Comments

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

Example

```
OUTP:EXT ON
```

Enable Trig Out port to output pulse after each scanned channel is closed.

:EXTernal[:STATe]?

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

Example

Query Trig Out Port Enable State

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

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

[:STATe]

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

Parameters

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

Comments

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

Example

Enabling Trig Out Port

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

[:STATe]?

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

Example

Query Trig Out Port Enable State

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

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

:TTLTrg[:STATe]

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

Parameters

Parameter Name	Parameter Type	Range of Values
<*n*>	numeric	0 to 7
<*mode*>	boolean	ON | OFF | 1 | 0

Comments

- **Enabling TTL Trigger Bus:** When enabled, a pulse is output from the selected TTL Trigger bus line (0 - 7) after each channel in the switchbox is closed. If disabled, a pulse is not output. The output is a negative going pulse.

- **One Output Selected at a Time:** Only one output (TTLTrg or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output.

- **Related Commands:** [ROUTe:]SCAN, TRIGger:SOURce, OUTPut:TTLTrg<*n*>[:STATe]?

- ***RST Condition:** OUTP:TTLT<*n*>:STAT 0

Example

Enabling TTLT Trigger Bus Line 7

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

:TTLTrg[:STATe]?

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

Example

Query TTL Trigger Bus Line Enable State

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

```
OUTP:TTLT7:STAT 1                  Enable TTL Trigger bus line 7.
OUTP:TTL7?                         Query bus enable state.
```

[ROUTe:]

The [ROUTe:] command subsystem controls switching and scanning operations for relay matrix modules.

Subsystem Syntax

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

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

CLOSe

[ROUTe:]CLOSe <*channel_list*> closes the relay matrix channels specified by *channel_list*. *channel_list* has the form (@ssrrcc) where ss = matrix card number (01-99), rr = matrix row number, and cc = matrix column number.

Parameters

Parameter Name	Parameter Type	Range of Values
<*channel_list*>	numeric	Agilent E1465A rr:00 - 15 cc:00 - 15 Agilent E1466A rr:00 - 03 cc:00 - 63 Agilent E1467A rr:00 - 07 cc:00 - 31

Comments

- **Closing Channels:**

  To close
  - a single channel, use [ROUTe:]CLOSe (@ssrrcc) ;
  - multiple channels, use [ROUTe:]CLOSe (@ssrrcc,ssrrcc,...) ;
  - sequential channels, use [ROUTe:]CLOSe (@ssrrcc:ssrrcc) ;
  - groups of sequential channels, use [ROUTe:]CLOSe (@ssrrcc:ssrrcc,ssrrcc:ssrrcc) ;
  - or any combination.

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

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

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

Example **Closing Matrix Channels**

This example closes channels 10100 and 20013 of a two-module switchbox (card numbers 01 and 02).

CLOS (@10100,20013)	*Closes row 01, column 00 of card #1 and row 00, column 13 of card #2.*

CLOSe?

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

Comments

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

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

Example **Query Channel Closure**

This example closes channels 10100 and 20013 of a two-module switchbox and queries channel closure. Since the channels are programmed to be closed 1,1 is returned as a string.

CLOS (@10100,20013)	*Closes row 01, column 00 of card #1, and column 13 of card #2.*
CLOS? (@10100,20013)	*Query channel closure.*

OPEN

[ROUTe:] OPEN <*channel_list*> opens the relay matrix channels specified by *channel_list*. *channel_list* has the form (@ssrrcc) where ss = matrix card number (01-99), rr = matrix row number, and cc = matrix column number.

Parameters

Parameter Name	Parameter Type	Range of Values
<*channel_list*>	numeric	Agilent E1465A rr: 00 - 15 cc: 00 - 15 Agilent E1466A rr: 00 - 03 cc: 00 - 63 Agilent E1467A rr: 00 - 07 cc: 00 - 31

Comments

- **Opening Channels:**

  To open
  – a single channel, use [ROUTe:]OPEN (@ssrrcc) ;
  – multiple channels, use [ROUTe:]OPEN (@ssrrcc,ssrrcc,...) ;
  – sequential channels, use [ROUTe:]OPEN (@ssrrcc:ssrrcc) ;
  – groups of sequential channels, use
  [ROUTe:]OPEN (@ssrrcc:ssrrcc,ssrrcc:ssrrcc) ;
  – or any combination.

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

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

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

Example

Opening Channels

This example opens channels 10100 and 20013 of a two-module switchbox (card numbers 01 and 02).

OPEN (@10100,20013)	*Open channels 10100 and 20013.*

OPEN?

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

Comments

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

Note

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

Example

Query Channel Open State

This example opens channels 10100 and 20013 of a two-module switchbox and queries channel 20013 state. Since channel 20013 is programmed to be open, `1` is returned.

OPEN (@10100,20013)	*Open channels 10100 and 20013.*
OPEN? (@20013)	*Queries channel opening.*

SCAN

[ROUTe:] SCAN <*channel_list*> defines the channels to be scanned. *channel_list* has the form (@ssrrcc) where ss = matrix card number (01-99), rr = matrix row number, and cc = matrix column number.

Parameters

Parameter Name	Parameter Type	Range of Values
<*channel_list*>	numeric	Agilent E1465A rr: 00 - 15 cc: 00 - 15 Agilent E1466A rr: 00 - 03 cc: 00 - 63 Agilent E1467A rr: 00 - 07 cc: 00 - 31

Comments

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

- **Scanning Channels:**
  You can scan
  – single channels (@ssrrcc) ;
  – multiple channels (@ssrrcc,ssrrcc,...) ;
  – sequential channels (@ssrrcc:ssrrcc) ;
  – groups of sequential channels (@ssrrcc:ssrrcc,ssrrcc:ssrrcc) ;
  – or any combination.

- **Scanning Operation:** When a valid channel list is defined, INITiate[:IMMediate] begins the scan and closes the first channel in the channel list. Successive triggers from the source specified by TRIGger:SOURce advances the scan through the channel list. At the end of the scan, the last trigger opens the last channel.

- **Stopping Scan:** See the ABORt command.

- **Related Commands:** TRIG:SOUR

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

Example

Scanning Using External Device

See “Scanning Channels” in Chapter 3 for example scanning programs using external instruments.

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.

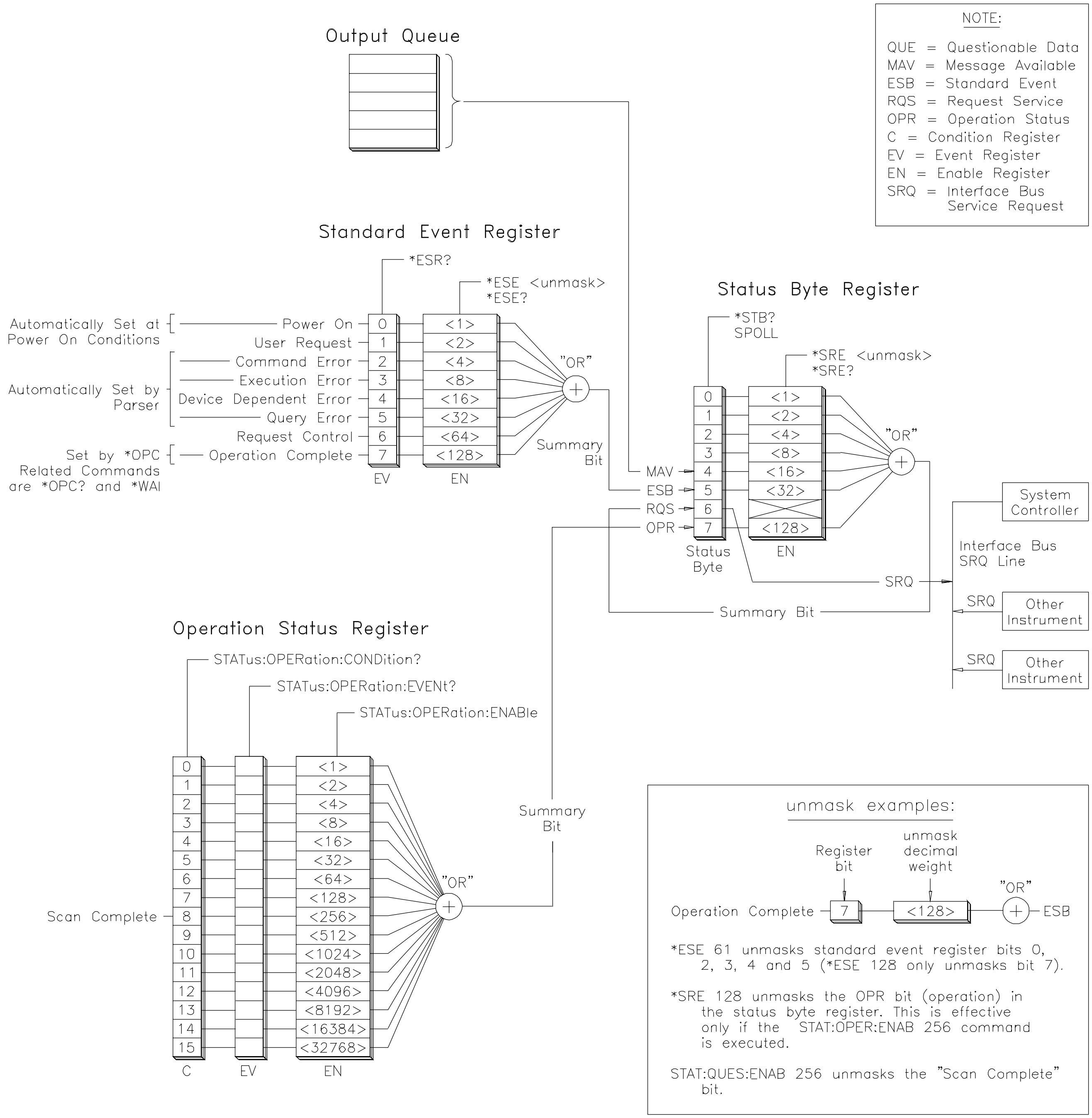
Subsystem Syntax

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

The STATus system contains four registers, two of which are under IEEE 488.2 control; the Event Status Register (*ESE?) and the Status Byte Register (*STB?). The Operational Status bit (OPR), Service Request bit (RQS), Event Summary bit (ESB), Message Available bit (MAV) and Questionable Data bit (QUE) in the Status Byte Register (bits 7, 6, 5, 4 and 3 respectively) can be queried with the *STB? command. Use the *ESE? command to query the *unmask* value for the 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 4-1.

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

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



NOTE:

QUE = Questionable Data
MAV = Message Available
ESB = Standard Event
RQS = Request Service
OPR = Operation Status
C = Condition Register
EV = Event Register
EN = Enable Register
SRQ = Interface Bus Service Request

*ESE 61 unmasks standard event register bits 0, 2, 3, 4 and 5 (*ESE 128 only unmasks bit 7).

*SRE 128 unmasks the OPR bit (operation) in the status byte register. This is effective only if the STAT:OPER:ENAB 256 command is executed.

STAT:QUES:ENAB 256 unmasks the "Scan Complete" bit.

Figure 4-1. Agilent E1465/66/67A Status System Register Diagram

:OPERation :CONDition?

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

:OPERation:ENABle

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

Parameters

Parameter Name	Parameter Type	Range of Values
<*unmask*>	numeric	0 through 65,535

Comments

- **Setting Bit 7 of the Status Byte Register:** STATus:OPERation:ENABle 256 sets bit 7 of the Status Byte Register to 1 after bit 8 of the Standard Operation Status Register is set to 1.

- **Related Commands:** [ROUTe:]SCAN

Example

Enabling Operation Status Register Bit 8

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

:OPERation:ENABle?

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

Comments

- **Output Format:** Returns a decimal weighted value from 0 to 65,535 indicating which bits are set to true.

- **Maximum Value Returned:** The value returned is the value set by the STAT:OPER:ENAB<*unmask*> command. However, the maximum decimal weighted value used in this module is 256 (bit 8 set to true).

Example

Query the Operation Status Enable Register

STAT:OPER:ENAB?	*Queries the Operation Status Enable Register.*

:OPERation[:EVENt]?

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

Comments

- **Setting Bit 8 of the Operation Status Register:** Bit 8 (Scan Complete) is set to 1 after a scanning cycle completes. Bit 8 returns to 0 after sending the STATus:OPERation[:EVENt]? command.

- **Returned Data After Sending the STATus:OPERation[:EVENt]? command:** The command returns `+256` if bit 8 of the Operation Status Register is set to 1. The command returns `+0` if bit 8 of the Operation Status Register is set to 0.

- **Event Register Cleared:** Reading the Event Register with the STATus:OPERation[:EVENt]? command clears it.

- **Aborting a Scan:** Aborting a scan will leave bit 8 set to 0.

- **Related Commands:** [ROUTe:] SCAN

Example

Reading the Operation Status Register After a Scanning Cycle

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

:PRESet

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

SYSTem

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

Subsystem Syntax

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

:CDEScription?

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

Parameters

Parameter Name	Parameter Type	Range of Values
<*number*>	numeric	1 to 99

Comments

- **Agilent E1465A Relay Matrix Module Description:** The SYSTem:CDEScription? <*number*> command returns:

```
16 x 16 Matrix Switch
```

- **Agilent E1466A Relay Matrix Module Description:** The SYST:CDEScription? <*number*> command returns:

```
4 x 64 Matrix Switch
```

- **Agilent E1467A Relay Matrix Module Description:** The SYST:CDEScription? <*number*> command returns:

```
8 x 32 Matrix Switch
```

Example

Reading the Description of a Card #l Module

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

:CTYPe?

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

Parameters

Parameter Name	Parameter Type	Range of Values
<*number*>	numeric	1 to 99

Comments

- **Agilent E1465A Relay Matrix Module Model Number:** The SYSTem:CTYPe? <*number*> command returns:

  ```
  HEWLETT-PACKARD,El465A,0,A.04.00
  ```

  where the `0` after E1465A is the module serial number (always 0) and `A.04.00` is an example of the module revision code number.

- **Agilent E1466A Relay Matrix Module Model Number:** The SYSTem:CTYPe? <*number*> command returns:

  ```
  HEWLETT-PACKARD,El466A,0,A.04.00
  ```

  where the `0` after E1466A is the module serial number (always 0) and `A.04.00` is an example of the module revision code number.

- **Agilent E1467A Relay Matrix Module Model Number:** The SYSTem:CTYPe? <*number*> command returns:

  ```
  HEWLETT-PACKARD,El467A,0,A.04.00
  ```

  where the `0` after E1467A is the module serial number (always 0) and `A.04.00` is an example of the module revision code number.

Example

Reading the Model Number of a Card #l Module

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

:CPON

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

Parameters

Parameter Name	Parameter Type	Range of Values
<*number*>	numeric	1 to 99 | ALL

Comments

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

Example

Setting Card #1 Module to its Power-On State

```
SYST:CPON 1                               Sets module #1 to power-on
                                          state.
```

:ERRor?

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

Comments

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

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

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

Example

Reading the Error Queue

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

TRIGger

The TRIGger command subsystem controls the triggering operation of relay matrix modules in a switchbox.

Subsystem Syntax

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

[:IMMediate]

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

Comments

- **Executing the TRIGger[:IMMediate] Command:** A channel list must be defined with [ROUTe:]SCAN <*channel_list*> and an INITiate[:IMMediate] command must be executed before TRIGger[:IMMediate] will execute.

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

- **Related Commands:** INITiate, [ROUTe:]SCAN

Example

Advancing Scan Using TRIGger Command

This example scans a single-module switchbox from channel 10000 through 10003. Since TRIGger:SOURce HOLD is set, the scan is advanced one channel each time TRIGger is executed.

```
TRIG:SOUR HOLD                  Sets trigger source to HOLD.
SCAN (@10000:10003)             Defines channel list.
INIT                            Begin scan, close channel 00.
loop statement                  Start count loop.
TRIG                            Advance scan to next channel.
increment loop                  Increment loop count.
```

:SOURce

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

Parameters

Parameter Name	Parameter Type	Parameter Description
BUS	discrete	*TRG or GET command
EXTernal	discrete	Trig In port
HOLD	discrete	Hold triggering
IMMediate	discrete	Immediate triggering
TTLTrg*n*	numeric	TTL Trigger Bus line *0* - *7*

Comments

- **Enabling the Trigger Source:** The TRIGger:SOURce command only selects the trigger source. The INITiate[:IMMediate] command enables the trigger source.

- **Using the TRIG Command:** You can use TRIGger[:IMMediate] to advance the scan when TRIGger:SOURce BUS or TRIGger:SOURce HOLD is selected.

- **Using External Trigger Inputs:** With TRIGger:SOURce EXTernal selected, only one switchbox at a time can use the external trigger input at the Agilent El406A Trig In port. The trigger input is assigned to the first switchbox that requested the external trigger source (with a TRIGger:SOURce EXTernal command).

- **Assigning External Trigger:** A switchbox assigned with TRIGger:SOURce EXTernal remains assigned to that source until the switchbox trigger source is changed to BUS, HOLD, or IMMediate. When the source is changed, the external trigger source is available to the next switchbox which requests it (with a TRIGger:SOURce EXTernal command). If a switchbox requests an external trigger input already assigned to another switchbox an error is generated.

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

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

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

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

Example

Scanning Using External Triggers

This example uses external triggering (TRIG:SOUR EXT) to scan channels 0000 through 0003 of a single-module switchbox. The trigger source to advance the scan is the input to the Trig In port on the Agilent E1406A Command Module. When INIT is executed, the scan is started and channel 0000 is closed. Then, each trigger received at the Trig In port advances the scan to the next channel.

TRIG:SOUR EXT	*Select external triggering.*
SCAN (@10000:10003)	*Scan channels 10000 through 10003.*
INIT	*Begin scan, close channel 10000.*
trigger externally	*Advance scan to next channel.*

Example

Scanning Using Bus Triggers

This example uses bus triggering (TRIG:SOUR BUS) to scan channels 0000 through 0003 of a single-module switchbox. The trigger source to advance the scan is the *TRG command (as set with TRIGger:SOURce BUS). When INIT is executed, the scan is started and channel 0000 is closed. Then, each *TRG command advances the scan to the next channel.

TRIG:SOUR BUS	*Select interface (bus) triggering.*
SCAN (@10000:10003)	*Scan channels 10000 through 10003.*
INIT	*Begin scan, close channel 10000.*
loop statement	*Loop to scan all channels.*
*TRG	*Advance scan using bus triggering.*
increment loop	*Increment loop count.*

:SOURce?

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

Example

Query Trigger Source

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

TRIG:SOUR EXT	*Set external trigger source*
TRIG:SOUR?	*Query trigger source*

IEEE 488.2 Common Commands

The following table lists the IEEE 488.2 Common (*) commands that apply to the relay matrix module. The operation of some of these commands is described in Chapter 3 of this manual. For more information on Common commands, refer to the *Agilent E1406 Command Module User's Manual* or the *ANSI/IEEE Standard 488.2-1987*.

Command	Title	Description
*IDN?	Identification Query	Returns identification string of the matrix module.
*RST	Reset	Opens all channel relays.
*TST?	Self-Test Query	Returns +0 if self-test passes. Returns +cc01 for firmware error. Returns +cc02 for bus error. Returns +cc10 if an interrupt was expected but not received. Returns +cc11 if the busy bit was not held for 7 msec.
*OPC	Operation Complete	Sets the Request for OPC flag when all pending operations have completed. Also sets OPC bit in the Standard Event Register.
*OPC?	Operation Complete Query	Returns a 1 to the output queue when all pending operations have completed. Used to synchronize between multiple instruments.
*WAI	Wait to Continue	Prevents an instrument from executing another command until the operation caused by the previous command is finished. Since all instruments normally perform sequential operations, this command causes no change to the instrument's operation.
*CLS	Clear Status Register	Clears all event registers, the Request for OPC flag, and all queues (except output query).
*ESE<*unmask*>	Event Status Enable	Used to set the bits in the Event Status Enable Register.
*ESE?	Event Status Enable Query	Queries the current contents in the Event Status Enable Register.
*ESR?	Event Status Register Query	Queries and clears current then contents in the Standard Event Status Register.
*SRE<*unmask*>	Service Request Enable	Used to set the Service Request Enable Register bits, and corresponding Serial Poll Status Register bits, to generate a service request.
*SRE?	Service Request Enable query	Queries the current contents in the Service Request Enable Register.
*STB?	Read Status Byte Query	Queries the current contents in the Status Byte Register.
*TRG	Trigger	Triggers the switchbox to advance the scan when scan is enabled and trigger source is TRIGger:SOURce BUS.
*RCL	Recall Saved State	Recalls previously stored matrix configuration.
*SAV	Save State	Stores the current matrix configuration in memory.

SCPI Command Quick Reference

The following table summarizes the SCPI commands for the matrix modules.

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

Notes

Appendix A
Specifications

General

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

Relay Life:
@ No load: 10^7 operations
@ Full load: 10^5 operations

Terminals:
Screw type, maximum wire size 18 AWG

Power Requirements:

Voltage:	**+5**	**+12 V**
Peak Module Current (A):	0.10	0.18

Watts/slot: 5 W

Cooling/slot: 0.08 mm H_2O @ 0.42 liter/sec

Operating Temperature: 0 - 55° C

Operating Humidity: 65% RH, 0 - 40° C

Input Characteristics

Maximum Voltage Terminal to Terminal:
200 Vdc; 170 Vac_{rms} (238 V ac peak to peak)

Maximum Voltage Terminal to Chassis:
200 Vdc 170 Vac_{rms} (238 V ac peak to peak)

Maximum Current per Channel (non-inductive):
1 Adc; 1A ac peak

Maximum Power per Channel:
30 Wdc; 62.5 VA ac resistive load

DC Performance

Thermal Offset per Channel:
<5µV (differential H-L)

Closed Channel Resistance:
Initial <4.0Ω
End of life <10.0Ω

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

AC Performance

Closed Channel Capacitance (Hi-Lo, Lo-Chassis):

Hi to Lo	<270 pF
Hi to GND	<430 pF
Lo to GND	<440 pF

Bandwidth (-3dB): Zload = Zsource = 50Ω:
>10 MHz

Crosstalk Between Channels:
See tables on next page.

Agilent E1465A AC Performance. Specifications are for 16 x 16 matrix, for Z(load) = Z(source) = 50 Ω. AC specifications apply with no more than one crosspoint closed per row or column.

Bandwidth (-3dB): >10MHz Crosstalk (dB)[1]

Within a Card (worst path)	**<10kHz**	**<100kHz**	**<1MHz**
Closed Path to Closed Path (typical)[3]	-78 dB	-57 dB	-41 dB
Open Row to Open Row (typical)[3]	-93 dB	-73 dB	-56 dB
Open Row to Open Column (typical)[3]	-84 dB	-63 dB	-47 dB
Open Column to Open Column (typical)[3]	-86 dB	-65 dB	-48 dB
Module to Module[2] (Represents 16 x 32 Configuration)			
Closed Path to Closed Path (typical)[3]	-78 dB	-58 dB	-43 dB
Open Row to Open Row (typical)[3]	-84 dB	-66 dB	-52 dB
Open Row to Open Column (typical)[3]	-84 dB	-63 dB	-48 dB
Open Column to Open Column (typical)[3]	-93 dB	-72 dB	-48 dB

Closed-Channel Capacitance

Hi to Lo:	<270pF
Hi to Ground:	<430pF
Lo to Ground:	<440pF

Agilent E1466A AC Performance. Specifications are for 4 x 64 matrix, for Z(load) = Z(source) = 50 Ω. AC specifications apply with no more than one crosspoint closed per row or column.

Bandwidth (-3dB): >10MHz Crosstalk (dB)[1]

Within a Card (worst path)	**<10kHz**	**<100kHz**	**<1MHz**
Closed Path to Closed Path (typical)[3]	-72 dB	-50 dB	-34 dB
Open Row to Open Row (typical)[3]	-73 dB	-52 dB	-37 dB
Open Row to Open Column (typical)[3]	-84 dB	-64 dB	-47 dB
Open Column to Open Column (typical)[3]	-92 dB	-70 dB	-52 dB
Module to Module[2] (Represents 4 x 128 Configuration)			
Closed Path to Closed Path (typical)[3]	-66 dB	-45 dB	-29 dB
Open Row to Open Row (typical)[3]	-68 dB	-46 dB	-29 dB
Open Row to Open Column (typical)[3]	-84 dB	-64 dB	-48 dB
Open Column to Open Column (typical)[3]	-92 dB	-71 dB	-52 dB

Closed-Channel Capacitance

Hi to Lo:	<270pF
Hi to Ground:	<430pF
Lo to Ground:	<440pF

Agilent E1467A AC Performance. Specifications are for 8 x 32 matrix, for Z(load) = Z(source) = 50 Ω. AC specifications apply with no more than one crosspoint closed per row or column.

Bandwidth (-3dB): >10MHz Crosstalk (dB)[1]

Within a Card (worst path)	**<10kHz**	**<100kHz**	**<1MHz**
Closed Path to Closed Path (typical)[3]	-75 dB	-54 dB	-38 dB
Open Row to Open Row (typical)[3]	-91 dB	-59 dB	-43 dB
Open Row to Open Column (typical)[3]	-85 dB	-64 dB	-47 dB
Open Column to Open Column (typical)[3]	-92 dB	-71 dB	-54 dB
Module to Module[2] (Represents 8 x 64 Configuration)			
Closed Path to Closed Path (typical)[3]	-72 dB	-51 dB	-33 dB
Open Row to Open Row (typical)[3]	-74 dB	-53 dB	-38 dB
Open Row to Open Column (typical)[3]	-92 dB	-72 dB	-56 dB
Open Column to Open Column (typical)[3]	-82 dB	-64 dB	-50 dB

Closed-Channel Capacitance

Hi to Lo	<270pF
Hi to Ground	<430pF
Lo to Ground	<440pF

NOTES: [1] Specifications are for worst crosspoint.

[2] Chaining cable used to connect modules (Agilent P/N E1466-80002).

[3] Typical is defined as the worst crosspoint test result from one or two matrix modules.

Relay Life

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

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

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

End-of-Life Detection

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

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

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

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

Replacement Strategy

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

Note

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

Notes

Appendix B
Matrix Registers

Using This Appendix

The Agilent E1465A/E1466A/E1467A matrix modules are register-based modules. When a SCPI command is sent to the matrix, the Agilent E1406A Command Module parses the command and programs the matrix at the register level.

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

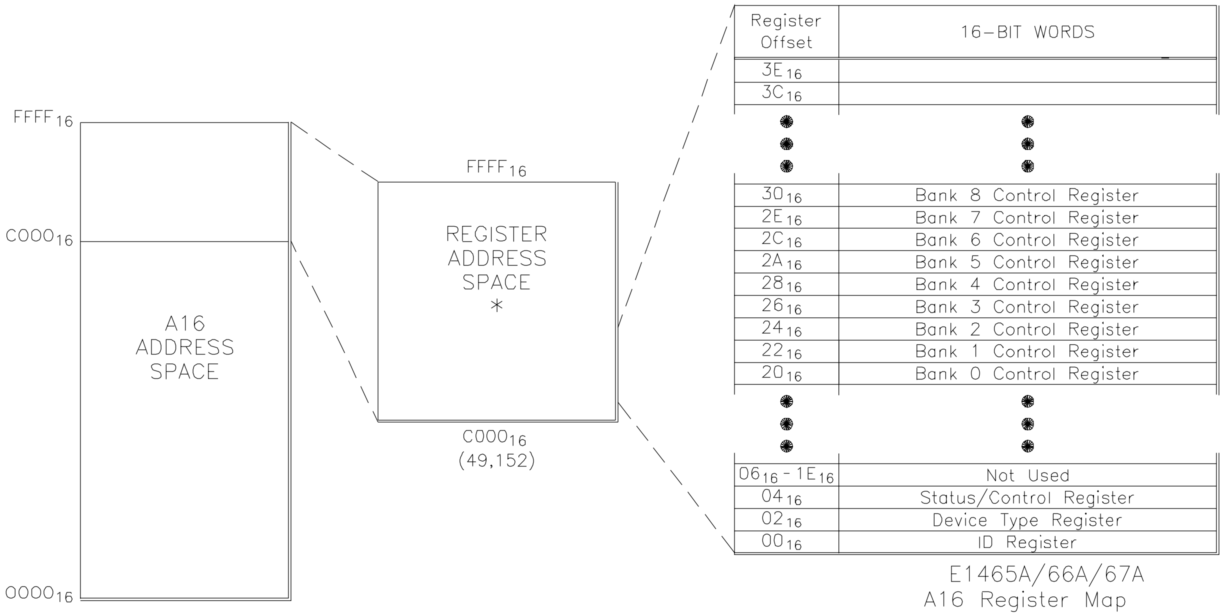
Addressing the Registers

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

Figure B-1 shows the register address location within A16 as it might be mapped by an embedded controller. Figure B-2 shows the location of A16 address space in the Agilent E1300/01 Mainframe and Agilent E1405/06A Command Module.

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

Register Address = Base Address + Register Offset

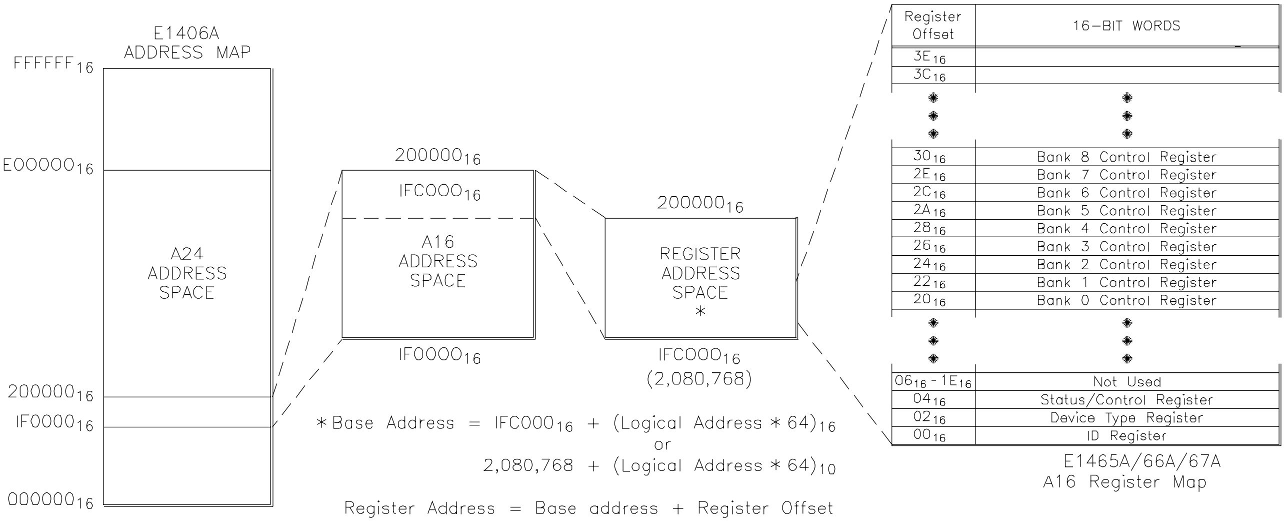


Register Offset	16-BIT WORDS
$3E_{16}$	
$3C_{16}$	
•	•
30_{16}	Bank 8 Control Register
$2E_{16}$	Bank 7 Control Register
$2C_{16}$	Bank 6 Control Register
$2A_{16}$	Bank 5 Control Register
28_{16}	Bank 4 Control Register
26_{16}	Bank 3 Control Register
24_{16}	Bank 2 Control Register
22_{16}	Bank 1 Control Register
20_{16}	Bank 0 Control Register
•	•
06_{16} - $1E_{16}$	Not Used
04_{16}	Status/Control Register
02_{16}	Device Type Register
00_{16}	ID Register

$*$Base Address = $C000_{16}$ + $(\text{Logical Address} * 64)_{16}$
or
49,152 + $(\text{Logical Address} * 64)_{10}$

Register Address = Base address + Register Offset

Figure B-1. Registers within A16 Address Space



Register Offset	16-BIT WORDS
$3E_{16}$	
$3C_{16}$	
•	•
30_{16}	Bank 8 Control Register
$2E_{16}$	Bank 7 Control Register
$2C_{16}$	Bank 6 Control Register
$2A_{16}$	Bank 5 Control Register
28_{16}	Bank 4 Control Register
26_{16}	Bank 3 Control Register
24_{16}	Bank 2 Control Register
22_{16}	Bank 1 Control Register
20_{16}	Bank 0 Control Register
•	•
06_{16} - $1E_{16}$	Not Used
04_{16}	Status/Control Register
02_{16}	Device Type Register
00_{16}	ID Register

$*$Base Address = $IFC000_{16}$ + $(\text{Logical Address} * 64)_{16}$
or
2,080,768 + $(\text{Logical Address} * 64)_{10}$

Register Address = Base address + Register Offset

Figure B-2. Registers within E1301/E1406 A16 Address Space

The Base Address

The base address used in register-based programming depends on whether the A16 address space is outside or inside the Agilent E1406A Command Module.

Address Space Outside the Command Module

When the command module is not part of your VXIbus system (Figure B-1), the matrix module's base address depends on the command module used:

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

Where $C000_{16}$ (49,152) is the starting location of the VXI A16 addresses, LADDR is the matrix module's logical address, and 64 is the number of address bytes per register-based module. For example, the matrix module's factory-set logical address is 120. If the address is not changed and an Agilent E1480A (V/360) Controller is used, the matrix module will have the following base address:

$C000_{16} + (120 * 64)_{16} = DE00_{16}$
or
49,152 + (120 * 64) = 56,832

Address Space Inside the Command Module

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

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

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

$1FC000_{16} + (120 * 64)_{16} = 1FDE00_{16}$
or
2,080,768 + (120 *64) = 2,088,448

Register Offset

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

$1FDE00_{16} + 04_{16} = 1FDE04_{16}$
or
2,088,448 + 04 = 2,088,452

Register Descriptions

The matrix modules contain 2 read registers, 1 read/write register, and 16 write registers. This section describes each matrix module register.

Reading and Writing to the Registers

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

- The Manufacturer's Identification Register (read only)
- The Device Type Register (read only)
- The Status/Control Register (read or write)
- Relay Control Registers (a total of 16 write-only registers)

Each of these registers is discussed in the following sections.

The Manufacturer's Identification Register

The manufacturer's identification register is at offset address 00_{16} and returns $\mathtt{FFFF}_{16}$. This shows Agilent as the manufacturer and that the module is an A16 register-based module. This register cannot be written to.

Manufacturer ID Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 00_{16}	FFFF_{16}															

The Device Identification Register

The device identification register is at offset address 02_{16} and returns $\mathtt{0122}_{16}$ if you have an Agilent E1465A/E1466A/E1467A Matrix Module. To determine which module you have, you must read the status/control register. The device identification register cannot be written to.

Device Identification Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 02_{16}	0122_{16}															

The Status/Control Register

The status/control register is at offset address 04_{16} and informs the user about the module's status and configuration. (See following example.)

Status/Control Register

Address Base + 04_{16}	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	X	**MS**	**Module ID**				X	X	**B**	**E**	X	X	1	1	X	X
Write	Not Used									**E**	Not Used					**SR**

Reading the Status/Control Register

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

Enable (bit 6)

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

Busy (bit 7)

0 indicates that the module is busy. Each relay requires about 7 msec execution time during which time the matrices are busy. Bit 7 of this register is used to inform the user of a busy condition.

Module ID (bits 10-13)

The following bit representations determine the module configuration (E1465A/66A/67A determined by the terminal card attached):

Module Identification

MODEL/BITS	(13)	(12)	(11)	(10)
E1465A	1	0	0	1
E1466A	0	1	1	0
E1467A	0	1	0	1

MODID Select (bit 14)

0 indicates that the module has been selected by MODID (Module ID), and 1 indicates it has not.

For example, if the Agilent E1466A matrix module is not busy (bit 7=1) and the interrupt is enabled (bit 6=0), then a read of the Status/Control Register (base + 04_{16}) returns DBBF.

Writing to the Status/Control Register

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

Soft Reset (bit 0)

Writing a 1 to this bit does not soft resets the module. To reset each relay in register-based programming, the programmer must write all 0's to all 16 banks to open all relays.

Enable(bit 6)

Writing a 1 to this bit disables the interrupt function of the module.

Note When writing to registers, 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. SCPI commands take care of this automatically.

The Relay Control Registers

There are 16 relay control registers. These registers are used to open and close the specified matrix relays. These registers are write registers, therefore when reading these registers, $FFFF_{16}$ is always returned.

The numbers shown in the register maps indicate the channel number to be written to. See Figures 1-1 through 1-3 to see the corresponding channel number. To close a relay you must write a 1 to the bit. For example,

WRITEIO -16,($DE00_{16}$ + 20_{16});0010_{16}

closes bit 4 of bank 0 (channel 004). Where $DE00_{16}$ is the base address, 20_{16} is the offset address, and 0010 is the hexadecimal number to send a 1 to bit 4.

See "Register-Based Programming Examples" later in this appendix for further examples.

- Bank 0 Relay Control Register (base + 20_{16})
- Bank 1 Relay Control Register (base + 22_{16})
- Bank 2 Relay Control Register (base + 24_{16})
- Bank 3 Relay Control Register (base + 26_{16})
- Bank 4 Relay Control Register (base + 28_{16})
- Bank 5 Relay Control Register (base + $2A_{16}$)
- Bank 6 Relay Control Register (base + $2C_{16}$)
- Bank 7 Relay Control Register (base + $2E_{16}$)
- Bank 8 Relay Control Register (base + 30_{16})
- Bank 9 Relay Control Register (base + 32_{16})
- Bank 10 Relay Control Register (base + 34_{16})
- Bank 11 Relay Control Register (base + 36_{16})
- Bank 12 Relay Control Register (base + 38_{16})
- Bank 13 Relay Control Register (base + $3A_{16}$)
- Bank 14 Relay Control Register (base + $3C_{16}$)

- Bank 15 Relay Control Register (base + $3E_{16}$)

Bank 0 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 20_{16}	015	014	013	012	011	010	009	008	007	006	005	004	003	002	001	000

Bank 1 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 22_{16}	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100

Bank 2 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 24_{16}	215	214	213	212	211	210	209	208	207	206	205	204	203	202	201	200

Bank 3 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 26_{16}	315	314	313	312	311	310	309	308	307	306	305	304	303	302	301	300

Bank 4 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 28_{16}	415	414	413	412	411	410	409	408	407	406	405	404	403	402	401	400

Bank 5 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + $2A_{16}$	515	514	513	512	511	510	509	508	507	506	505	504	503	502	501	500

Bank 6 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + $2C_{16}$	615	614	613	612	611	610	609	608	607	606	605	604	603	602	601	600

Bank 7 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + $2E_{16}$	715	714	713	712	711	710	709	708	707	706	705	704	703	702	701	700

Bank 8 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 30_{16}	815	814	813	812	811	810	809	808	807	806	805	804	803	802	801	800

Bank 9 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base + 32_{16}	915	914	913	912	911	910	909	908	907	906	905	904	903	902	901	900

Bank 10 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base+34_{16}	1015	1014	1013	1012	1011	1010	1009	1008	1007	1006	1005	1004	1003	1002	1001	1000

Bank 11 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base+36_{16}	1115	1114	1113	1112	1111	1110	1109	1108	1107	1106	1105	1104	1103	1102	1101	1100

Bank 12 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base+38$_{16}$	1215	1214	1213	1212	1211	1210	1209	1208	1207	1206	1205	1204	1203	1202	1201	1200

Bank 13 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base+3A$_{16}$	1315	1314	1313	1312	1311	1310	1309	1308	1307	1306	1305	1304	1303	1302	1301	1300

Bank 14 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base+3C$_{16}$	1415	1414	1413	1412	1411	1410	1409	1408	1407	1406	1405	1404	1403	1402	1401	1400

Bank 15 Relay Control Register

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base+3E$_{16}$	1515	1514	1513	1512	1511	1510	1509	1508	1507	1506	1505	1504	1503	1502	1501	1500

Register-Based Programming Examples

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

- Mainframe: Agilent 75000 Series C (Agilent E1400B)
- Controller: Agilent V/360 (Agilent E1480A) w/Resource Manager and Slot 0
- Programming Language: BASIC/UNIX, C
- Switch Card: Agilent Matrix Module (Agilent E1466A)

Reading the Registers

The following examples read the module ID, Device ID, and Status registers on the matrix module.

BASIC

```
10 !******************************************************************************
20 !****                              READREG                              *****
30 !******************************************************************************
40  OPTION BASE 1
50  ! Set up arrays to store register names and addresses
60  DIM Reg_name$(1:3)[32], Reg_addr(1:3)
70  !
80   ! Read register names and addresses into the arrays
90  READ Reg_name$(*)!Read in register names
100 READ Reg_addr(*) !Read in register addresses
110 !
120 ! Set base address variable
130 Base_addr = DVAL("DE00",16)
140 !
150 ! Map the A16 address space in the V/360
160 !
170 CONTROL 16,25;2
180 ! Call the subprogram Read_regs
190 Read_regs(Base_addr, Reg_name$(*), Reg_addr(*))
200 !
210 DATA Identification register, Device register, Status register
220 DATA 00, 02, 04
230 END


300 ! This subprogram steps through a loop that reads each register and
     !prints its contents
310 !
320 SUB Read_regs(Base_addr, Reg_name$(*), Reg_addr(*))
330 !
340 For Number = 1 to 3
350   Register = READIO(-16,Base_addr + Reg_addr(number))
360   PRINT Reg_name$(number); " = "; IVAL$(Register,16)
370 Next Number
380 SUBEND
```

UNIX, C

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

#define logical_address 120             /*logical address of the matrix module*/

int fd;
typedef unsigned short word;
typedef struct dev_regs      {                          /*set up pointers*/
                             unsigned short id_reg;
                             unsigned short device_type;
                             unsigned short status_reg;
                             unsigned short dummy[13];
                             unsigned short bank0_channels;
} DEV_REGS;

main()
{

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

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

}                                               /*END of main program*/

/*SUB READ_REG*/

int   read_reg(reg_ptr)
DEV_REGS *reg_ptr;
{

printf("\n ID Register = 0x%x\n",reg_ptr->id_reg);          /*read the id register*/


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

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

Making Measurements

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

You must insert your own programming code for the measurement part of this program. If you are using the Agilent E1411B, see the *Agilent E1326B/E1411B 5 1/2-Digit Multimeter User’s Manual* for programming examples.

BASIC

```
10  !****************************************************************************
20  !****                          MAKEMEAS                                *****
30  !****************************************************************************
40  OPTION BASE 1
50  !Set up arrays to store register names and addresses
60  DIM Reg_name$(1:1)[32], Reg_addr(1:1)
70  !
80  !Read register names and address into the arrays
90  READ Reg_name$(*)
100 READ Reg_addr(*)
110 !
120 !Set base address variable
130 Base_addr = DVAL("DE00",16)
140 !
150 !Map the A16 address space in the V/360
160 CONTROL 16,25;2
170 !Call the subprogram Make_meas
180 Make_meas(Base_addr, Reg_addr(*))
190 !
200 DATA Bank0 channels register
210 DATA 32
220 END

280 ! This subprogram closes bit 1 of bank0 channels, waits for the channel
    ! to be closed, makes a measurement, and then opens the relay.

300 SUB Make_meas(Base_addr, Reg_addr(*))
310 !
320 WRITEIO -16, Base_addr + Reg_addr(1); 1
330 REPEAT
340 UNTIL BIT(READIO(-16,Base_addr+4),7)
          .
          .     ! Make measurements
          .
380 WRITEIO -16, Base_addr + Reg_addr(1);0
390 SUBEND
```

UNIX, C

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

#define logical_address 120       /*logical address of the matrix module*/

int fd;
int j;
typedef unsigned short word;
typedef struct dev_regs  {                    /*set up pointers*/
                         unsigned short id_reg;
                         unsigned short device_type;
                         unsigned short status_reg;
                         unsigned short dummy[13];
                         unsigned short bank0_channels;
} DEV_REGS;

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

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

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

/*SUB VER_TIME*/

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

struct timezone tzp;
```

Continued on next page.

```
gettimeofday(&first,&tzp);
for (j=0; j<=7000; j ++);
gettimeofday ($second,&tzp);

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

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

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

/*SUB MAKE_MEAS*/

int make_meas(reg_ptr)
DEV_REGS *reg_ptr;
{
                                            /*close bit 1 of bank0 */
reg_ptr->bank0_channels=0x0001;
for (j=0; j<=7000; j ++);                   /*wait for switch to close*/
printf("\n Making Measurement");
        .
        .                                   /*make measurements*/
        .
                                            /*open bit 1 of bank0*/
reg_ptr->bank0_channels=0x0000;
return;
}
```

Note

The sub **ver_time** allows time for switch closures. This sub should print a time around 7 ms. If the time is less, you must change the value of **j** in the **for** loop. For example, instead of 7000, you might have to use 10000.

Scanning Channels

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

Again, you must insert your own programming code for the measurement part of this program. If you are using the Agilent E1411B, see the *Agilent E1326B/E1411B 5 1/2-Digit Multimeter User's Manual* for programming examples.

BASIC

```
10 !******************************************************************************
20 !****                    SCANNING                                         *****
30 !******************************************************************************
40  OPTION BASE 1
50  !Set up arrays to store register names and addresses
60  DIM Reg_name$(1:1)[32], Reg_addr(1:1)
70  !
80  !Read register names and addresses into the arrays
90  READ Reg_name$(*)
100 READ Reg_addr(*)
110 !Set base address variable
120 Base_addr = DVAL("DE00",16)
130 !
140 !Map the A16 address space in the V/360
150 CONTROL 16,25;2
160 !Call the subprogram Scan_meas
170 Scan_meas(Base_addr, Reg_addr(*))
180 !
190 DATA Bank0 channels register
200 DATA 32
210 END

280 ! This subprogram sets all bits in bank0 open then scan through bank 0,
    ! closing one channel at a time (waits for the channel to be closed) so a !
    !measurement can be made.

300 SUB Scan_meas(Base_addr, Reg_addr(*))
310 !
320 WRITEIO -16, Base_addr + Reg_addr(1);0
330 FOR I= 0 to 15
340                     WRITEIO -16, Base_addr + Reg_addr(1);2^I
350                     REPEAT
360                     UNTIL BIT(READIO(-16,Base_addr+4),7)
370     PRINT "Making Measurement"
            .
            .    ! Make measurements
            .
420 NEXT I
430 WRITEIO -16,Base_addr + Reg_addr(1);0
440 SUBEND
```

UNIX,C

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

#define logical_address 120  /*logical address of the matrix module*/
#define lastch   15

int fd;
int i;
int j;
int  reg;
double y;
typedef unsigned short word;
typedef struct dev_regs  {                /*set up pointers*/
                  unsigned short id_reg;
                  unsigned short device_type;
                  unsigned short status_reg;
                  unsigned short dummy[13];
                  unsigned short bank0_channels;
} DEV_REGS;
main()
{
                                /*open the Agilent V/360 VXI interface*/
fd=open("/dev/vxi/primary",O_RDWR);
if (fd)       {
   perror("open");
   exit(1);
 }                                   /*retrieve the A16 pointers*/
dev=(struct dev_regs *)vxi_get_a16_addr(fd,logical_address);

ver_time();               /*sub to verify the time to close the switch*/
           /*sub to close a set of switches and make measurements*/

scan_meas(dev);
}                                    /*END of main program*/

/*SUB VER_TIME*/

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

Continued on next page.

```
struct timezone tzp;

gettimeofday(&first,&tzp);
for (j=0; j<=7000; j ++);
gettimeofday ($second,&tzp);
if (first.tv_usec > second.tv_usec)
        {
        second.tv_usec +=1000000;
        second.tv_sec--;
}

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

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

int scan_meas(reg_ptr)
DEV_REGS *reg_ptr;
{
reg_ptr->bank0_channels=0x000;                    /*set bank0 to 000 */
i=0;
for (i=0;i=lastch;i ++)
        {
        y=i;
        reg=pow(2.0,y);
        reg_ptr-bank0_channels=reg;
        for (j=0; j<=7000; j ++);                 /*wait for switch to be closed*/
        printf("\n Making Measurement");
                .
                .                                 /*make measurements*/
                .
        }
return;
}
```

Note The sub **ver_time** allows time for switch closures. This sub should print a time around 7 ms. If the time is less, you must change the value of **j** in the **for** loop. For example, instead of 7000, you might have to use 10000.

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

Notes

Appendix C
Matrix Error Messages

The table below lists the error messages associated with the matrix modules programmed with SCPI commands. See the *Agilent E1406 Command Module User's Manual* for complete information on Error Messages.

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

Notes

Index

Agilent E1465/66/67A Relay Matrix Switch Modules