Agilent Technologies E8482A/B Six/Three 1x4 RF Multiplexer Module User's Manual

Agilent Technologies

Manual Part Number: E8482-90001
Printed in U.S.A. E0301

Contents

Agilent E8482A/B User's Manual

Notes:

AGILENT TECHNOLOGIES WARRANTY STATEMENT

AGILENT PRODUCT: E8482A/B Six/Three 1x4 RF Multiplexer Module **DURATION OF WARRANTY:** 3 years

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

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

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

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

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

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

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

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

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

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

U.S. Government Restricted Rights

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

Agilent Technologies

E8482A/B Six/Three 1x4 RF Multiplexer Module User's Manual
Edition 1

Documentation History

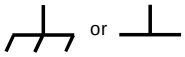
All Editions and Updates of this manual and their creation date are listed below. The first Edition of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct or add additional information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

Edition 1 . . . . . March, 2001

Safety Symbols

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

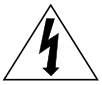
Indicates the field wiring terminal that must be connected to earth ground before operating the equipment — protects against electrical shock in case of fault.



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

Alternating current (AC)

Direct current (DC).



Warning. Risk of electrical shock.

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

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

WARNINGS

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

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

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

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

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

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

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

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

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

Manufacturer's Name: Agilent Technologies, Inc.
Manufacturer's Address: *Basic, Emerging and Systems Technologies Product Generation Unit*
815 14th Street S.W.
Loveland, CO 80537 USA

Declares, that the product

Product Name: Six/Three 1x4 RF Multiplexer Module
Model Number: E8482A/B
Product Options: *This declaration includes all options of the above product(s).*

Conforms with the following European Directives:

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

Conforms with the following product standards:

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

Supplemental Information:

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

September 5, 2000
Date

Name

Quality Manager
Title

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

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

Notes:

Chapter 1
Getting Started

About This Chapter

This chapter describes the Agilent E8482A RF Multiplexer (6 x 4:1) and the E8482B Multiplexer (3 x 4:1) modules, contains information on how to program them using SCPI (Standard Commands for Programmable Instruments) commands, and provides an example program to check initial operation. Chapter contents are:

RF Multiplexer Modules Description

The E8482A RF Multiplexer module is a VXIbus C-Size register-based product and can operate in a C-Size VXIbus mainframe. It is comprised of six 4:1 multiplexers which provide bidirectional switching for user inputs and outputs. If fewer channels are desired, order E8482B which consists of three 4:1 multiplexers only.

For the RF multiplexers, switching consists of connecting a channel to its common terminal. Only one channel in each bank can be and should be connected to its common terminal at a time.

In addition, up to two E1473A or E1475A Expander module can be connected to an E8482A or an E8482B RF Multiplexer module, providing a total of six or three 3 GHz 4:1 multiplexer banks and twelve 1.3 GHz 4:1 multiplexer banks. See “Expanding the RF Multiplexer” on page 24 of this manual for details.

NOTE *The E8482B module is identical in operation to the E8482A module with the exception of fewer channels (three 4:1 multiplexers) included. Unless otherwise stated, this manual applies to both modules.*

NOTE *The E8482A module is compatible with the E1472A module. If you would like to replace your existing E1472A modules with the newly designed E8482A modules without changing any source code of your application programs, you may have to do some configuration for the module. Refer to “Setting the E1472A Mode Switch” on page 26 of this manual for details.*

Simplified Schematic

As shown in Figure 1-1, the E8482A 3GHz RF Multiplexer module consists of six banks of channels (banks 0 - 5) to form six 4:1 multiplexers and the E8482B module consists of three banks of channels (banks 0 - 2) to form three 4:1 multiplexers. The switching sections of each bank are identical. See Figure 1-1 for a simplified switching diagram. Banks are arranged as follows:

- Bank 0 includes channels 00 through 03 and Com 00.
- Bank 1 includes channels 10 through 13 and Com 10.
- Bank 2 includes channels 20 through 23 and Com 20.
- Bank 3 includes channels 30 through 33 and Com 30.
- Bank 4 includes channels 40 through 43 and Com 40.
- Bank 5 includes channels 50 through 53 and Com 50.

In the remainder of this manual, channels are referred as *n*0 through *n*3 and Com *n*0, where *n* is the bank number (0-5 for E8482A and 0-2 for E8482B). Each channel is switched (connected to its common) by closing the appropriate (nonlatching) relays. Channels *n*0 through *n*3 can be switched to Com *n*0 for all banks. Only one channel in each bank can be connected to its common at a time.

User inputs/outputs to each channel are via SMB connectors on the front panel. When a channel is CLOSed, it is internally connected to its COMmon connector. When a channel is open, it is internally disconnected. Open channels are not terminated. At power-off, power-on, or reset, channel *n*0 is connected to the COM *n*0 connector, and all other channels (*n*1 through *n*3) are open (non-terminated) for all banks.

NOTE *The E8482B consists of three banks of channels (banks 0-2). Banks 0-5 are applicable for the E8482A module.*

Typical Configuration

The RF Multiplexer relays are configured in a "tree" structure which provides high isolation and low VSWR (voltage standing wave ratio). Each channel can switch user inputs up to 30 Vdc or 30 Vac peak. User input frequencies to the 50 Ω RF Multiplexer module can be from DC to 3 GHz. The RF Multiplexer can be configured for several arrangements, such as standard, tree, or matrix (see Chapter 3 of this manual for more information).

For a Standard Commands for Programmable Instruments (SCPI) environment, one or more RF Multiplexers (with or without Expanders connected) can be defined as a switchbox instrument. For a switchbox instrument, all RF Multiplexer channels within the instrument can be addressed using a single interface address.

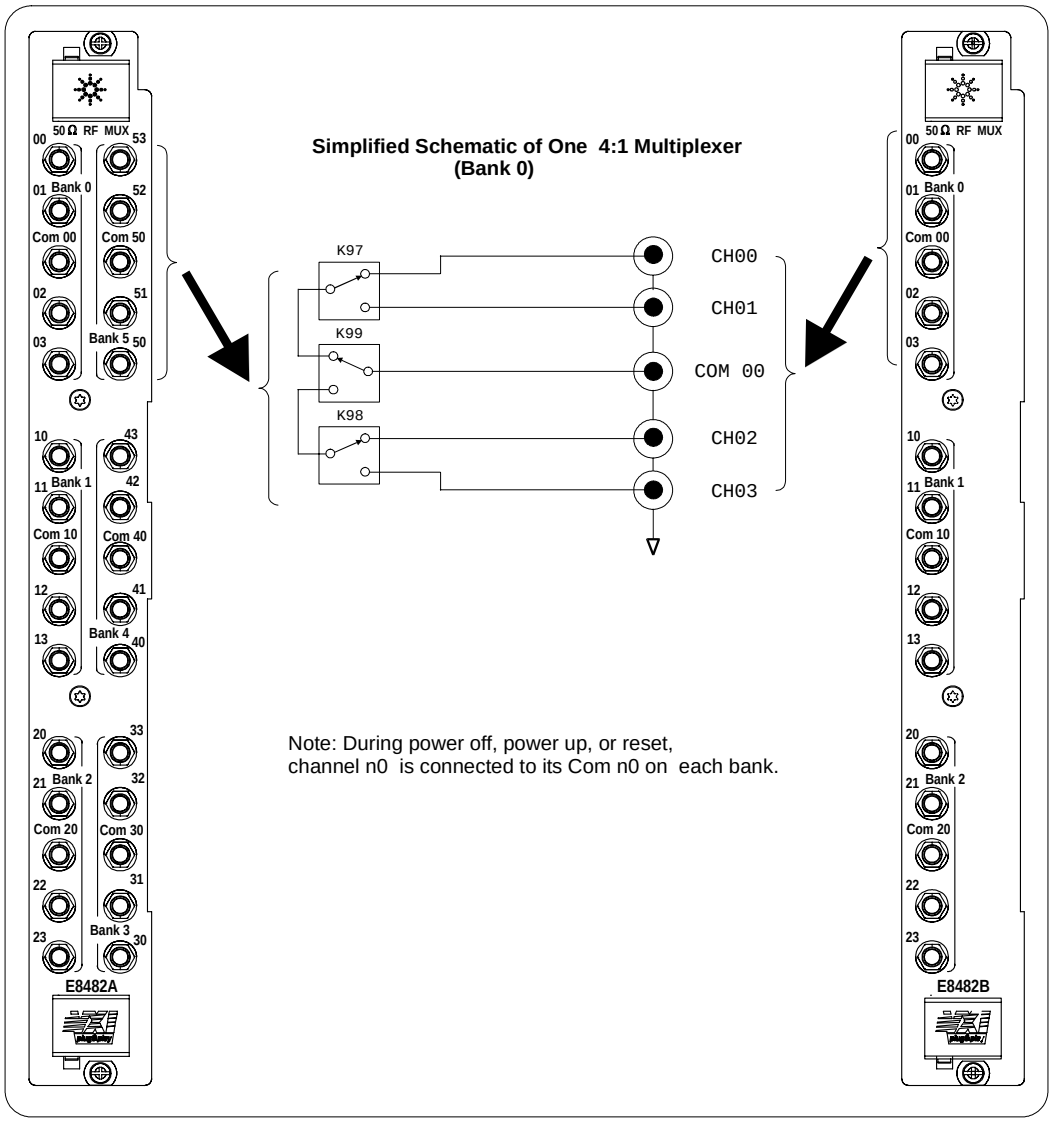


Figure 1-1. Multiplexer Switching Diagram

Instrument Definition

The plug-in modules installed in an Agilent 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 Multimeter Instrument).

Programming the RF Multiplexer

To program the RF Multiplexer using SCPI commands, you must select the controller language, interface address, and SCPI commands to be used. See the *C-Size VXIbus System Configuration Guide* for detailed interface addressing and controller language information. For uses in other systems or mainframes, see the appropriate manuals. For more details of SCPI commands applicable to the module, refer to Chapter 4 of this manual.

NOTE *This discussion below applies to SCPI programming. The module can also be programmed by directly writing to its registers. See Appendix B, "RF Multiplexer Registers" for details on registers programming.*

Specifying SCPI Commands

To address specific channels within an E8482A/B RF Multiplexer, you must specify the SCPI command and RF Multiplexer channel address. For example, use CLOSe <*channel_list*> to switch (connect) channels. Refer to Chapter 4 of this manual for a complete list of SCPI commands applicable for the RF Multiplexer.

Channel Addresses

Only valid channel addresses can be included in a *channel_list*. For the RF Multiplexer, the channel address has the form of (@*ccmmnn*) where

cc = RF Multiplexer module card number (01-99)
mm = RF Multiplexer/Expander module number (00-02)
nn = channel number

To specify a *channel_list*, use the form of:

- (@ccmmnn) for a single channel
- (@ccmmnn,ccmmnn) for multiple channels
- (@ccmmnn:ccmmnn) for sequential channels
- (@ccmmnn:ccmmnn,ccmmnn:ccmmnn) for groups of sequential channels
- or any combination of the above.

Card Number

The card number (*cc* of the *channel_list*) identifies the RF Multiplexer module within the switchbox. The RF Multiplexer card number depends on the switchbox configuration (single-module or multiple-module) set for the RF Multiplexer. Leading zeroes can be ignored for the card number.

NOTE *The Expander module(s) card number is the same as the RF Multiplexer it is connected to.*

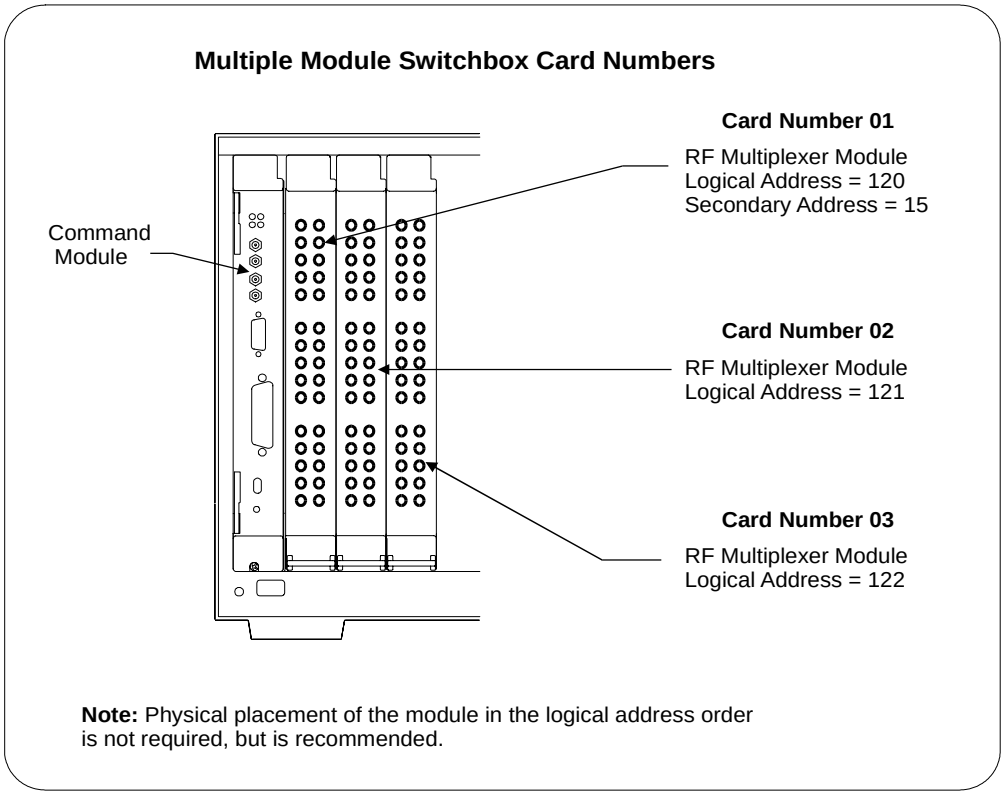


Figure 1-2. Card Numbers for Multiple-Module Switchbox

- **Single-module**. For a single-module switchbox, the card number is always 01.

- **Multiple-module**. For a multiple-module switchbox, the RF 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 1-2 illustrates the card numbers and logical addresses of a typical multiple-module switchbox in an Agilent C-Size Mainframe with an Agilent command module. For uses in other systems or mainframes, see the appropriate manuals.

Module Number

The module number (*mm* of the *channel_list*) applies ONLY when an RF Multiplexer has one or more E1473A Expander modules attached. Figure 1-3 illustrates a typical example for the module numbers.

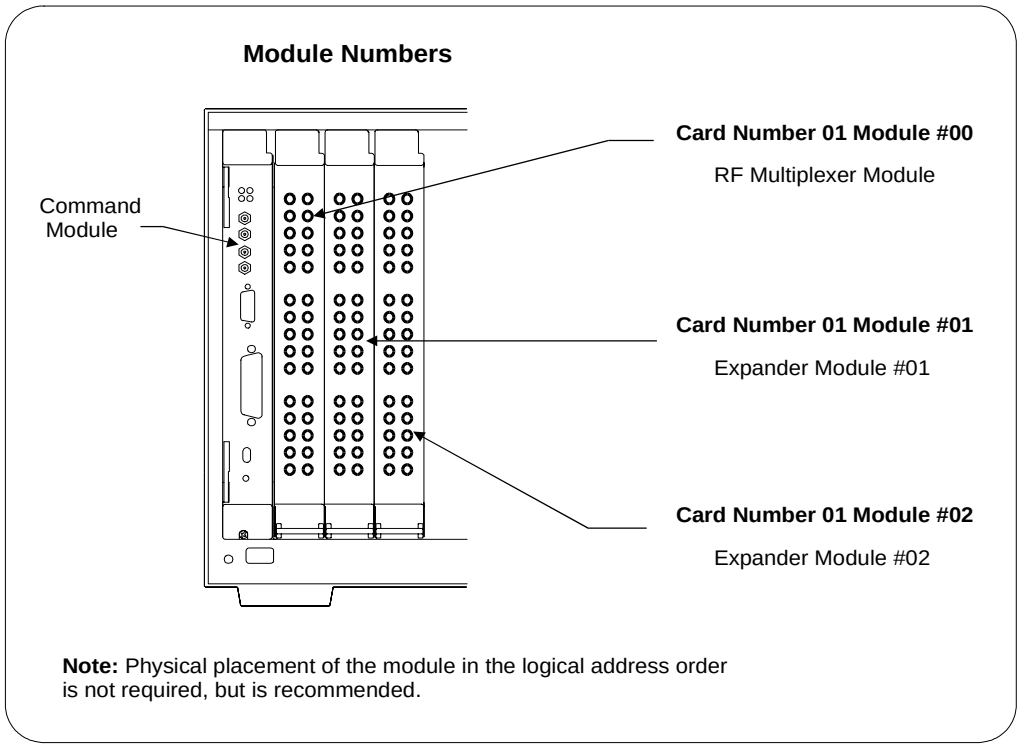


Figure 1-3. Module Numbers

- RF Multiplexer. Module number is 00. If the RF Multiplexer does not have Expander module(s) connected, the module number can be omitted.
- Expander module #1. Module number is 01. Used to select the Expander module (if installed) connected to the RF Multiplexer RMD 3/RMD 2 internal ribbon connector.
- Expander module #2. Module number is 02. Used to select the Expander module (if installed) connected to the RF Multiplexer RMD 5/RMD 4 internal ribbon connector.

Channel Number

The channel number (*nn* of the *channel_list*) identifies the channel within an RF module. The E8482A RF multiplexer has banks 0-5 and the E8482B has banks 0-2. The channels for each bank are numbered as:

- channels 00 through 03 for Bank 0
- channels 10 through 13 for Bank 1
- channels 20 through 23 for Bank 2
- channels 30 through 33 for Bank 3
- channels 40 through 43 for Bank 4
- channels 50 through 53 for Bank 5

The RF Multiplexers will accept and execute channel ranges (ccmmnn:ccmmnn) without generating an error, but the result is to close the last channel in each bank within the range specified. For example, after CLOSe (@10101:10151) is executed, card 01, module 01, channels 03, 13, 23, 33, 43, and 51 would remain closed.

Example: RF Multiplexer Channel List

```
CLOS (@ 10001)          ! Connect channel 01 to Com 00
                          on the RF Multiplexer module,
                          card #1.
CLOS (@ 10101, 20101)   ! Connect channel 01 to Com 00
                          on the Expander #1 module,
                          card #01 and card #02.
```

Initial Operation

Use the following example programs to perform the initial operation on the E8482A RF Multiplexer module. To run the programs, an Agilent E1406A command module is required. Also, you must download the E8482A SCPI driver into the E1406A command module and have the SICL Library, the VISA extensions, and an Agilent 82350 GPIB card installed and properly configured in your PC.

In the examples, an E8482A RF Multiplexer module without Expander modules connected, is defined as a single-multiplexer switchbox instrument. The computer interfaces to the mainframe via GPIB. The GPIB interface select code is 7, the GPIB primary address is 09, and the E8482A module is at logical address 120 (secondary address = 120/8 = 15). Refer to the *Agilent E1406A Command Module User's Guide* for more addressing information. For more details on the related SCPI commands used in the examples, see Chapter 4 of this manual.

Example: Closing a Channel (HTBasic)

This example program was written in HTBasic programming language. The program closes channel 02 of the RF Multiplexer module, then queries the channel closure state. The result is returned to the computer and displayed on the screen (1 = channel closed, 0 = channel open).

```
10  DIM Ch_Stat$[20]          ! Dimension a variable.
20  OUTPUT 70915; "*RST"      ! Resets the module. Switches
                                all channel n0 to Com n0.
30  OUTPUT 70915; "CLOS (@102)"  ! Connect channel 02
                                   to Com 00.
40  OUTPUT 70915; "CLOS? (@102)" ! Query channel 02
                                   closure state.
50  ENTER 70915;Ch_Stat$      ! Enter results into value.
60  PRINT Ch_Stat$            ! Display results. "1" returned
                                indicates channel 02 is closed.
70  END
```

Example: Closing a Channel (C/C++)

This example program was developed and tested in Microsoft® Visual C++ 6.0 but should compile under any standard ANSI C compiler. The program closes channel 02 of the RF Multiplexer module, then queries the channel closure state. The result is returned to the computer and displayed on the screen (1 = channel closed, 0 = channel open).

```
#include <visa.h>
#include <stdio.h>
#include <stdlib.h>

    /* Module logical address is 120, secondary address is 15 */
#define INSTR_ADDR "GPIB0::9::15::INSTR"

int main()
{
   ViStatus errStatus;                  /* Status from each VISA call */
   ViSession viRM;                      /* Resource manager session */
   ViSession E8482A;                    /* Module session */
   char ch_stat[10];                    /* Channel state */

    /* Open the default resource manager */
   errStatus = viOpenDefaultRM (&viRM);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpenDefaultRM() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Open the module instrument session */
   errStatus = viOpen(viRM,INSTR_ADDR, VI_NULL,VI_NULL,&E8482A);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpen() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Reset the module */
   errStatus = viPrintf(E8482A, "*RST;*CLS\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Close channel 02 of card 1*/
   errStatus = viPrintf(E8482A, "ROUT:CLOS (@102)\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Query closure state of channel 102 */
   errStatus = viQueryf(E8482A,"ROUT:CLOS? (@102)\n","%t",ch_stat);
   if (VI_SUCCESS > errStatus) {
      printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
      return errStatus;}
   printf("Channel 102 state is: %s\n",ch_stat);
```

```
    /* Close the module instrument session */
    errStatus = viClose (E8482A);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viClose() returned 0x%x\n",errStatus);
        return 0;}

    /* Close the resource manager session */
    errStatus = viClose (viRM);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viClose() returned 0x%x\n",errStatus);
        return 0;}

    return VI_SUCCESS;
}
```

Notes:

Chapter 2
Configuring the RF Multiplexer

About This Chapter

This chapter shows how to configure the RF Multiplexer module for use in a VXIbus mainframe, install it in a mainframe, and connect external wiring to the RF Multiplexer module. Chapter contents include:

Warnings and Cautions

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

CHANNEL WIRING INSULATION. All channels that have a common connection must be insulated so that the user is protected from electrical shock in the event that two or more channels are connected together. This means wiring for all channels must be insulated as though each channel carries the voltage of the highest voltage channel.

Caution **MAXIMUM POWER. The maximum power that can be applied to any SMB connector is 10 W or 10 VA. The maximum voltage that can be applied to any SMB connector is 30 Vdc or 30 Vac peak. The maximum current that can be applied to any SMB connector is 0.5 Adc.**

STATIC ELECTRICITY. Static electricity is a major cause of component failure. To prevent damage to the electrical components in the RF Multiplexer, observe anti-static techniques whenever removing a module from the mainframe or whenever working on a module.

Setting the Logical Address Switch

The logical address switch (LADDR) factory setting is 120. Valid address values are from 1 to 255. Figure 2-1 shows the logical address switch position and setting 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.*

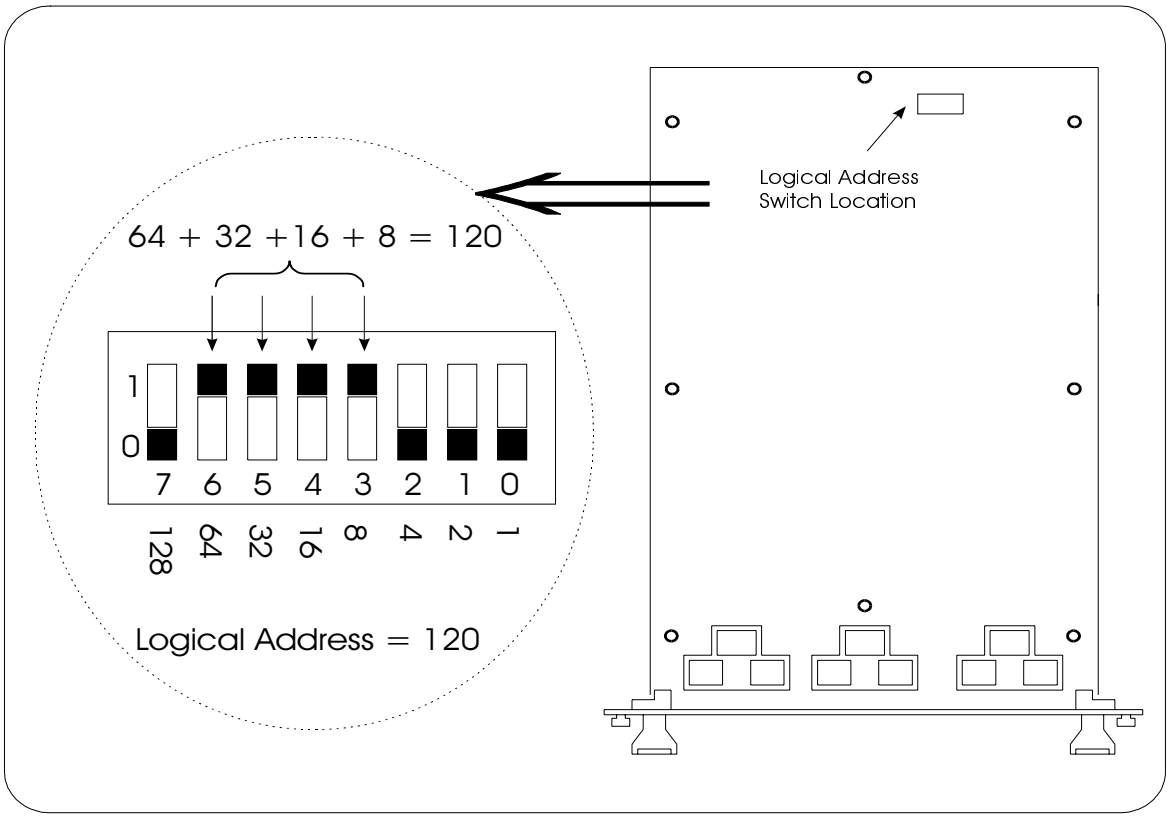


Figure 2-1. Setting the Logical Address Switch

Setting the Interrupt Priority

The RF Multiplexer module generates an interrupt after a channel has been closed. These interrupts are sent to, and acknowledgments are received from, the command module (Agilent E1406A) via the VXIbus backplane interrupt lines.

For most applications where the RF Multiplexer module is installed in an Agilent 75000 Series C mainframe, the interrupt priority level does not have to be changed. This is because the VXIbus interrupt lines have the same priority and interrupt priority is established by installing modules in slots numerically closest to the command module. Thus, slot 1 has a higher priority than slot 2, slot 2 has a higher priority than slot 3, etc.

By default, the interrupt priority level is Level 1. It can be set to any one of the VXI backplane lines 1-7 (corresponding to Levels 1-7) either by sending SCPI or directly writing to the Interrupt Selection Register. Level 1 is the lowest priority and Level 7 is the highest priority. The interrupt can also be disabled at power-up, after a SYSRESET, or by sending SCPI or directly writing to the Status/Control Register. See Page 44 of this manual for the details of the related SCPI commands. For more information about register writing, see Page 69 on *Appendix B, E8482A Register-Based Programming* for more information.

NOTE *Changing the interrupt priority level is not recommended. DO NOT change it unless specially instructed to do so. Refer to the E1406A Command Module User's Manual for more details.*

NOTE *When the E8482A is set to the E1472A mode, you CANNOT change the interrupt priority level with SCPI or by directly writing to the Interrupt Selection Register. See Page 26 of this manual on how to set the interrupt priority level when in E1472A mode.*

Expanding the RF Multiplexer

The E8482A RF Multiplexer module is capable of controlling up to two E1473A Expander modules, providing a total of eighteen (4:1) banks. The Expander modules can be physically located in the C-Size mainframe next to the RF Multiplexer, or up to eight meters away using expansion cables. Locating the Expander module close to the external device keeps connecting cable lengths to a minimum, thereby reducing the possibility of crosstalk and insertion loss of high frequency signals.

Use Figure 2-2 and the procedure below to connect the Expander Modules:

1. Verify a 3-1 cable (P/N E1472-61601) is installed in the Remote Module Driver (RMD) 2/4/5 cable slot. Cable can be connected without removing the shield.

2. Cut the cable ties holding the cables, and pull the cables through the slots in the shield.

3. Mark the 3-1 cables as shown in Figure 2-2 (cross-out unused RMD number).

4. Connect the 3-1 cables to the Expander Modules as follows: RMD3 cable to the Expander (module 01) Bank 3-5 connector. RMD2 cable to the Expander (module 01) Bank 0-2 connector. RMD5 cable to the Expander (module 02) Bank 3-5 connector. RMD4 cable to the Expander (module 02) Bank 0-2 connector.

NOTE *RMD1 cable is connected to the E8482A RF Multiplexer Bank 3-5 connector, and RMD0 cable is connected to the E8482A Bank 0-2 connector at the factory. These connections are not accessible with the shield in place and should not be moved.*

5. If the Expander module is physically located (up to eight meters) away from the mainframe, you can daisy chain up to 10 extender cables (P/N E1473-80002) for each RMD connection (see Figure 2-2, module 02).

6. Fold and tie unused RMD cables.

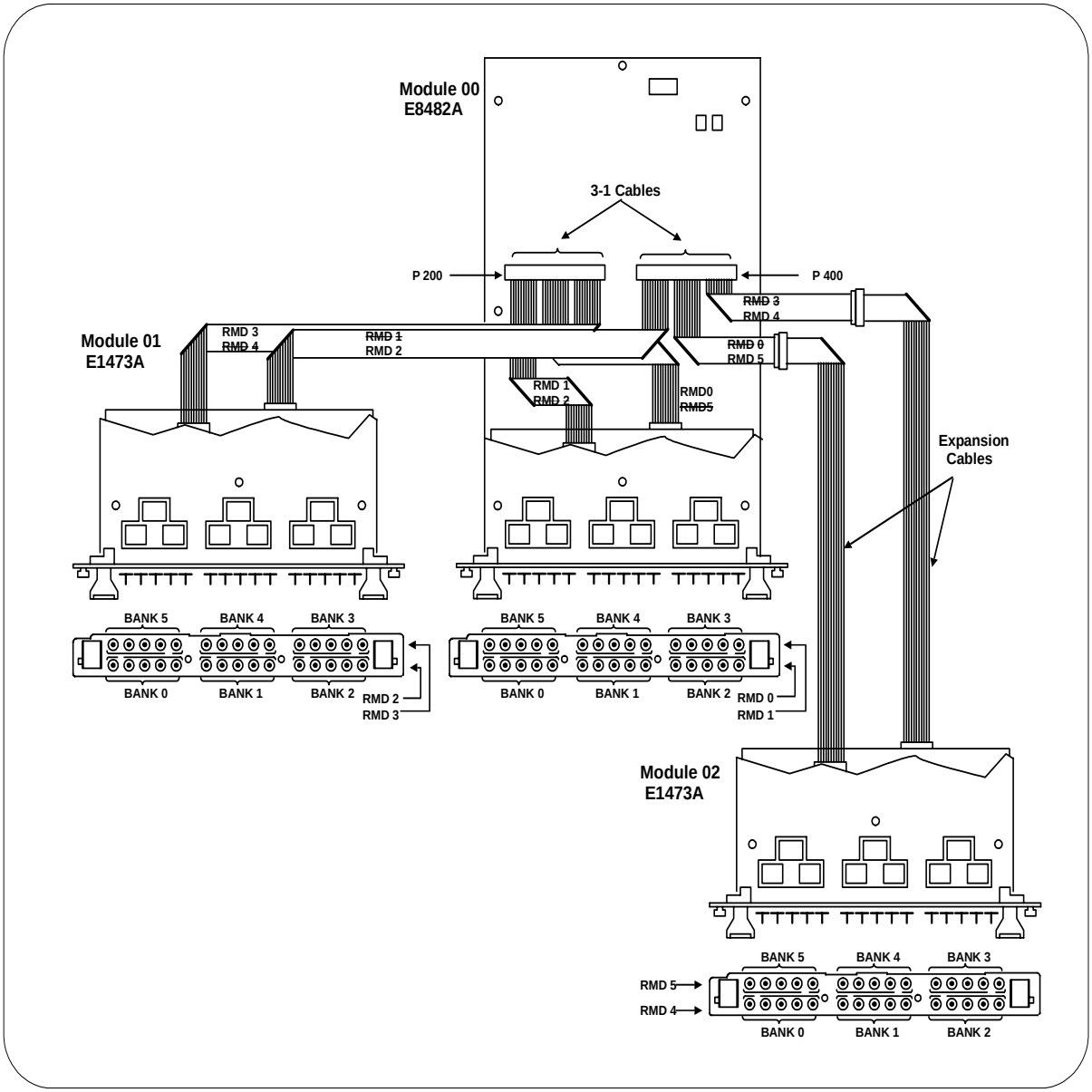
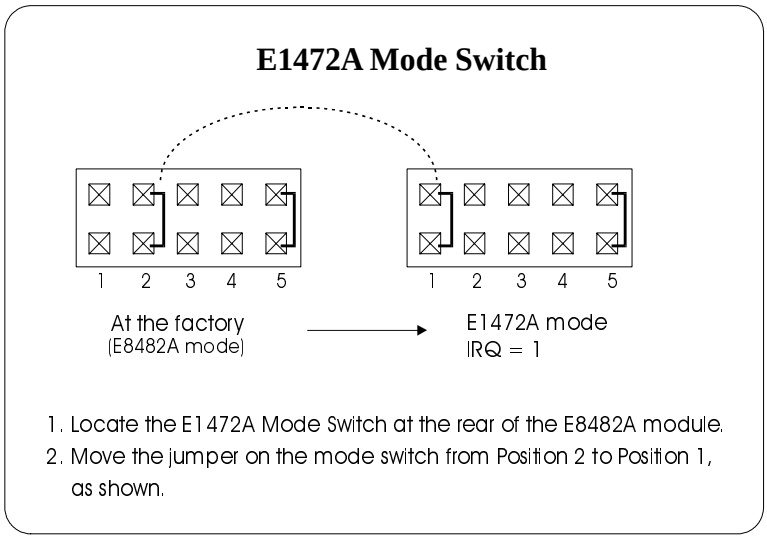


Figure 2-2. Expander Module Connection

Setting the E1472A Mode Switch

The E8482A RF Multiplexer module is compatible with the E1472A 1.3 GHz RF Multiplexer module. If you want to replace the E1472A modules with the newly designed E8482A modules without changing any source code of your existing application programs, you should configure the E1472A Mode Switch on the E8482A module. Figure 2-3 shows the E1472A Mode Switch position and setting information.



Position Numbers (1=Jumper; 0=No Jumper; - =Unused) 1 2 3 4 5	Jumper Configuration
0 - 0 0 1	E8482A mode (factory default)
1 - 0 0 0	E1472A mode, IRQ = X
1 - 0 0 1	E1472A mode, IRQ = 1
1 - 0 1 0	E1472A mode, IRQ = 2
1 - 0 1 1	E1472A mode, IRQ = 3
1 - 1 0 0	E1472A mode, IRQ = 4
1 - 1 0 1	E1472A mode, IRQ = 5
1 - 1 1 0	E1472A mode, IRQ = 6
1 - 1 1 1	E1472A mode, IRQ = 7

NOTES:

a. Position 1 is used for setting E1472A/E8482A mode.
1 = E1472A mode; 0 = E8482A mode.
b. Position 2 is not used.
c. Postions 3 - 5 are used for setting Interrupt Priority Level when in E1472A mode and are not used in E8482A mode.

Figure 2-3. Setting the E1472A Mode Switch

NOTE *The E1472A Mode Switch does not apply for the E8482B module.*

NOTE *DO NOT change the factory default setting for the E1472A Mode Switch unless you want the module to work in the E1472A mode.*

NOTE *When shipped from the factory, no jumper is installed on the positions 1 of the E1472A Mode Switch indicating the module defaults to the E8482A mode. To change to the E1472A mode, simply remove the jumper from the position 2, then reinstall it to the position 1.*

NOTE *When in the E1472A mode, you CANNOT change the interrupt priority level with SCPI or by directly writing to the Interrupt Selection Register. However, you can disable/enable the interrupt by writing to the Status/Control Register as shown on Page 66 of this manual.*

NOTE *When in the E1472A mode, you can select seven different interrupt levels 1-7 by installing jumper(s) on the corresponding positions 3-5 of the E1472A Mode Switch (as shown in Figure 2-3). A jumper is installed on the position 5 of the E1472A Mode Switch at the factory, which indicates the factory setting is Level 1. Level X should not be used under normal operating conditions. Changing the priority level jumper is not recommended. DO NOT change unless specially instructed to do so.*

Connecting User Inputs to the Module

User inputs to the E8482A/B RF Multiplexer and E1473A Expander modules are made through user-supplied female 50Ω SMB connectors, while to the E1475A Expander modules are through 75Ω SMB connectors. Figure 2-4 shows the E8482A and the E8482B male SMB connectors and associated channel numbers.

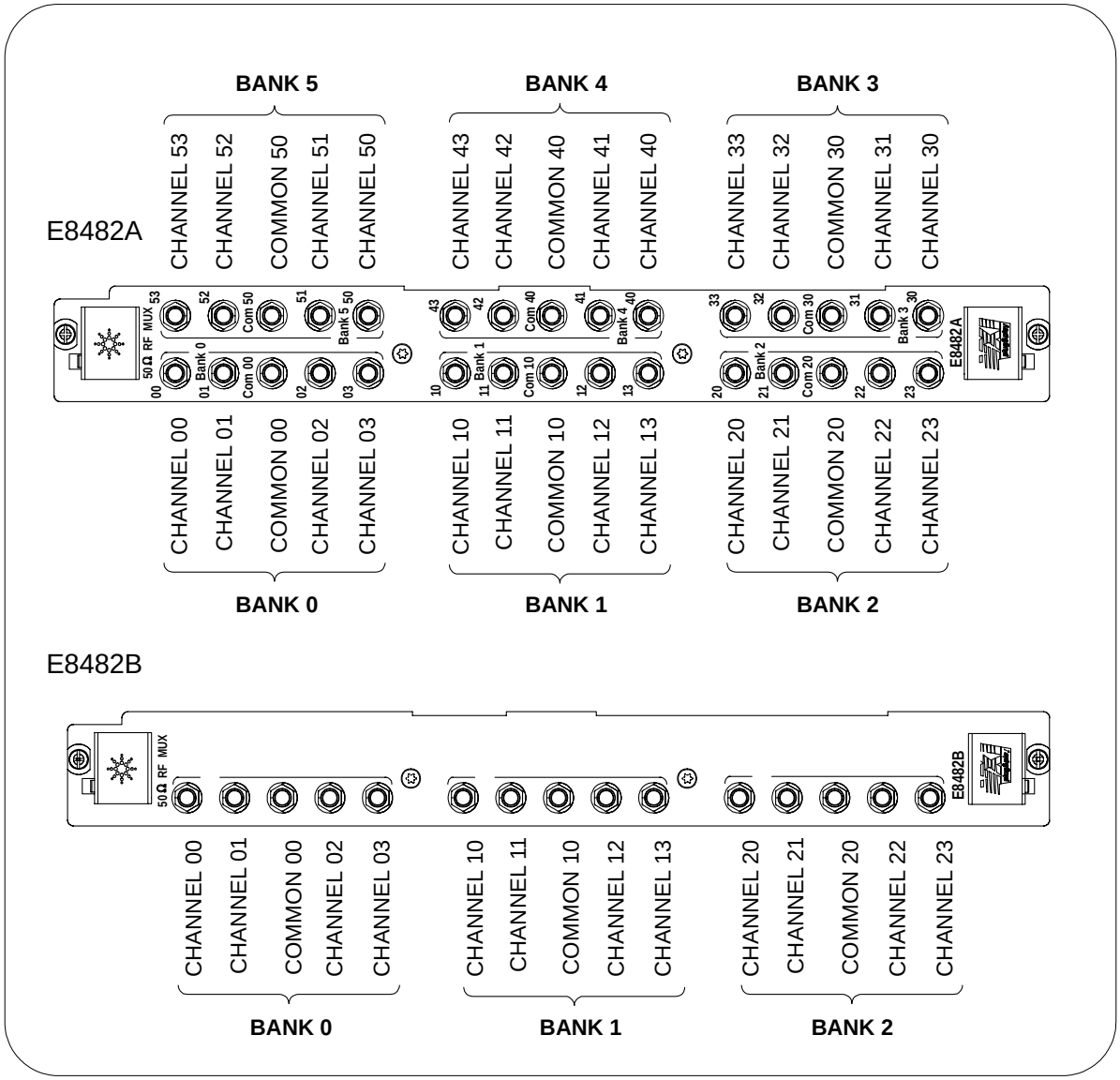


Figure 2-4. E8482A/B SMB Connectors Location

Cabling Guidelines

- For best high-frequency performance, user cabling to the 50 Ω connectors should have at least two braid shields or one braid and a foil wrap. However, the 75 Ω connectors (E1475A module) only work with one braid shield or one braid and a foil wrap.
- Always use shielded coaxial cables with the desired characteristic impedance. Keep cables as short as possible, especially in high-frequency circuits or pulse circuits where a rise/fall time is critical.
- Long cables can add delay time which can cause timing problems. All test equipment, such as counters, spectrum analyzer, and oscilloscopes must be terminated in the characteristic impedance (50 Ω or 75 Ω) to minimize reflection loss.

Notes:

Chapter 3
Using the RF Multiplexer

About This Chapter

This chapter uses typical examples to show how to use the RF Multiplexer and Expander modules. For the details of the commands used in this chapter, see Chapter 4, "RF Multiplexer Command Reference" of this manual. Chapter contents are:

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

RF Multiplexer Commands

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

Table 3-1. RF Multiplexer Commands Used in this Chapter

Commands	Description
[ROUTe:]CLOSe <*channel_list*>	Close the channels in the channel list.
[ROUTe:]CLOSe? <*channel_list*>	Query the state of the channels in the channel list.
[ROUTe:]OPEN? <*channel_list*>	Query the state of the channels in the channel list.
*RST	Connect channel n0 to COM n0 on all banks.
*CLS	Clear all switchbox status registers.
*STB?	Query status byte register

Power-On and Reset Conditions

When the RF Multiplexer is powered off, powered on, or *RST (reset), all banks close channel *n*0 to COM *n*0, where *n* = 0-5 for E8482A and *n* = 0-2 for E8482B.

Module Identification

The following example programs use the *RST, *CLS, *IDN?, SYST:CTYP?, and SYST:CDES? commands to reset and identify the RF Multiplexer module.

Example: Identifying Module (HTBasic)

```
10   DIM A$[50], B$[50], C$[50]          ! Dimension three string
                                           variables.
20   OUTPUT 70915; "*RST; *CLS"          ! Reset the module and clear
                                           Status Registers.
30   OUTPUT 70915; "*IDN?"               ! Query module identification.
40   ENTER 70915; A$                     ! Enter the result into A$.

50   OUTPUT 70915; "SYST:CDES? 1"        ! Query for module description.
60   ENTER 70915; B$                     ! Enter the result into B$.

70   OUTPUT 70915; "SYST:CTYP? 1"        ! Query for module type.
80   ENTER 70915; C$                     ! Enter the result into C$

90   PRINT A$, B$, C$                    ! Print the contents of the
                                           variable A$, B$ and C$.

100  END
```

Example: Identifying Module (C/C++)

```
#include <visa.h>
#include <stdio.h>
#include <stdlib.h>

    /* Module logical address is 120, secondary address is 15 */
#define INSTR_ADDR "GPIB0::9::15::INSTR"

int main()
{
   ViStatus errStatus;                  /* Status from each VISA call */
   ViSession viRM;                      /* Resource manager session */
   ViSession E8482A;                    /* Module session */
   char id_string[256];                 /* ID string */
   char m_desp[256];                    /* Module description */
   char m_type[256];                    /* Module type */

    /* Open the default resource manager */
   errStatus = viOpenDefaultRM (&viRM);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpenDefaultRM() returned 0x%x\n",errStatus);
      return errStatus;}
```

```
    /* Open the module instrument session */
    errStatus = viOpen(viRM,INSTR_ADDR, VI_NULL,VI_NULL,&E8482A);
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viOpen() returned 0x%x\n",errStatus);
        return errStatus;}

    /* Reset the module */
    errStatus = viPrintf(E8482A,"*RST;*CLS\n");
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
        return errStatus;}

    /* Query the module ID string */
    errStatus = viQueryf(E8482A,"*IDN?\n", "%t", id_string);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
        return errStatus;}
    printf("ID is %s\n", id_string);

    /* Query the module description */
    errStatus = viQueryf(E8482A,"SYST:CDES? 1\n", "%t", m_desp);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
        return errStatus;}
    printf("Module Description is %s\n", m_desp);

    /* Query the module type */
    errStatus = viQueryf(E8482A,"SYST:CTYP? 1\n", "%t", m_type);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
        return errStatus;}
    printf("Module Type is %s\n", m_type);

    /* Close the module instrument session */
    errStatus = viClose (E8482A);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viClose() returned 0x%x\n",errStatus);
        return 0;}

    /* Close the resource manager session */
    errStatus = viClose (viRM);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viClose() returned 0x%x\n",errStatus);
        return 0;}

    return VI_SUCCESS;
}
```

Switching Channels

For general purpose switch operation, you can connect a signal by closing a specific channel to its COMmon. Only one channel per bank can be closed at a time. The following channel configurations are possible.

- Close channels 00 through 03 to COM 00
- Close channels 10 through 13 to COM 10
- Close channels 20 through 23 to COM 20
- Close channels 30 through 33 to COM 30
- Close channels 40 through 43 to COM 40
- Close channels 50 through 53 to COM 50

Use CLOSe <*channel_list*> to close a channel to its COMmon. The <*channel_list*> has the form of (@ccmmnn) for a single channel, and (@ccmmnn,ccmmnn,...) for two or more channels. See "Channel Addresses" on page 14 for more details.

cc = card number (01-99)
mm = module number (00-02)
nn = channel number (00-03, 10-13, 20-23, 30-33, 40-43, 50-53)

NOTE *If the RF Multiplexer does not have Expander module(s) connected, the module number (mm) can be omitted.*

NOTE *The following examples are shown using multiple configurations (multiple-module switchbox and single-module switchbox with Expander modules) to illustrate programming differences. It is important that the user understand that all the examples shown could have been performed using only one E8482A RF Multiplexer module.*

Switching configurations include standard, matrix, and tree.

Example: Standard Switching

Use standard switching to switch channels *n*0 - *n*3 to COM *n*0 (where *n* = 0-5 for E8482A, or *n* = 0-2 for E8482B). One channel per bank can be connected to its common at a time.

This example connects channel 11 to COM 10 of the RF Multiplexer in a standard configuration as shown in Figure 3-1. The RF Multiplexer, without Expander modules connected, is defined as a single-multiplexer switchbox instrument.

The following example programs were written in HTBasic and C/C++ programming languages respectively. They will show you how to connect COM 10 to channel 11 by closing channel 111, then query to see whether it is closed. The result is returned to the computer and displayed (1 = channel closed, 0 = channel open).

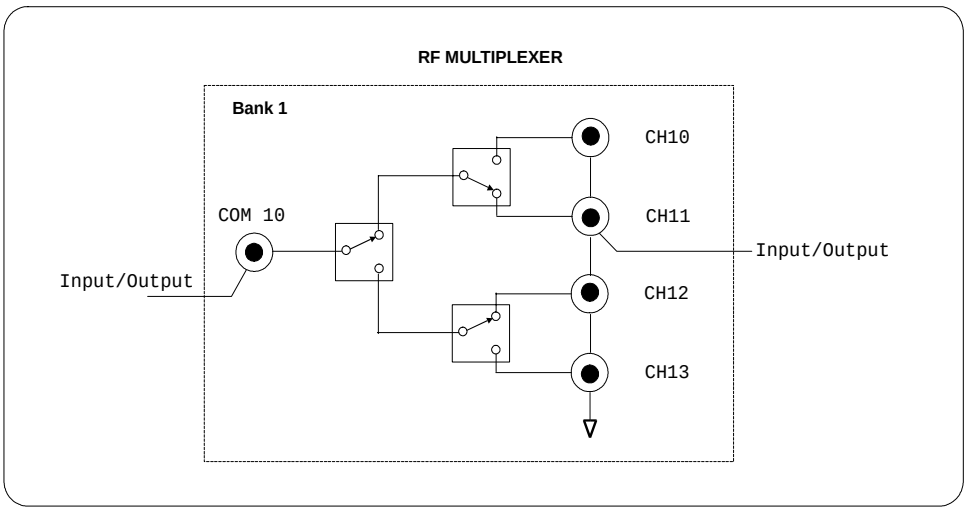


Figure 3-1. Example: RF Multiplexer Standard Switching

Programming with HTBasic

```
10  DIM A$[20]                              ! Dimension a string variable.
20  OUTPUT 70915; "*RST; *CLS"              ! Reset the module and clear
                                              Status Register.
30  OUTPUT 70915; "ROUT:CLOS (@111)" ! Close channels 111.
40  OUTPUT 70915; "ROUT:CLOS? (@111)"
                                            ! Query to see whether channel
                                              111 is closed.
50  ENTER 70915; A$                        ! Enter the result into A$.
60  PRINT A$                               ! "1" returned indicates it is
                                              closed.
70  END
```

Programming with C/C++

```
#include <visa.h>
#include <stdio.h>
#include <stdlib.h>

    /* Module logical address is 120, secondary address is 15 */
#define INSTR_ADDR "GPIB0::9::15::INSTR"

int main()
{
   ViStatus errStatus;                  /* Status from each VISA call*/
   ViSession viRM;                      /* resource manager session */
   ViSession E8482A;                    /* module session */
   char ch_stat[10];                    /* channel state*/

    /* Open the default resource manager */
   errStatus = viOpenDefaultRM (&viRM);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpenDefaultRM() returned 0x%x\n",errStatus);
      return errStatus;}
```

```
  /* Open the module instrument session */
errStatus = viOpen(viRM,INSTR_ADDR, VI_NULL,VI_NULL,&E8482A);
if(VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

  /* Reset the module */
errStatus = viPrintf(E8482A, "*RST;*CLS\n");
if(VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

  /* Close channel 11 of card 1*/
errStatus = viPrintf(E8482A, "CLOS (@111)\n");
if(VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

  /* Query closure state of channel 11 */
errStatus = viQueryf(E8482A,"ROUT:CLOS? (@111)\n", "%t",ch_stat);
if (VI_SUCCESS > errStatus) {
   printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
   return errStatus;}
printf("Channel 111 state is: %s\n",ch_stat);

  /* Close the module instrument session */
errStatus = viClose (E8482A);
if (VI_SUCCESS > errStatus) {
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

  /* Close the resource manager session */
errStatus = viClose (viRM);
if (VI_SUCCESS > errStatus) {
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}
return VI_SUCCESS;
}
```

Example: Tree Switching

Use tree switching to provide signal routing while maintaining characteristic impedance. With tree switching, signal delay time is more than doubled since the signal must pass through two or more channel banks plus extra cabling. Keep cables as short as possible, especially between channel banks, to minimize delay.

This example uses two RF Multiplexers in a tree configuration to connect COM 10 of RF Multiplexer #1 to channel 13 of RF Multiplexer #2. The two RF Multiplexers form a multiple-multiplexer switchbox instrument. As shown in Figure 3-2, to connect COM 10 to channel 13, execute: CLOS (@111,213). The programs for this example can be written by referring to those provided for “Example: Standard Switching” on page 34.

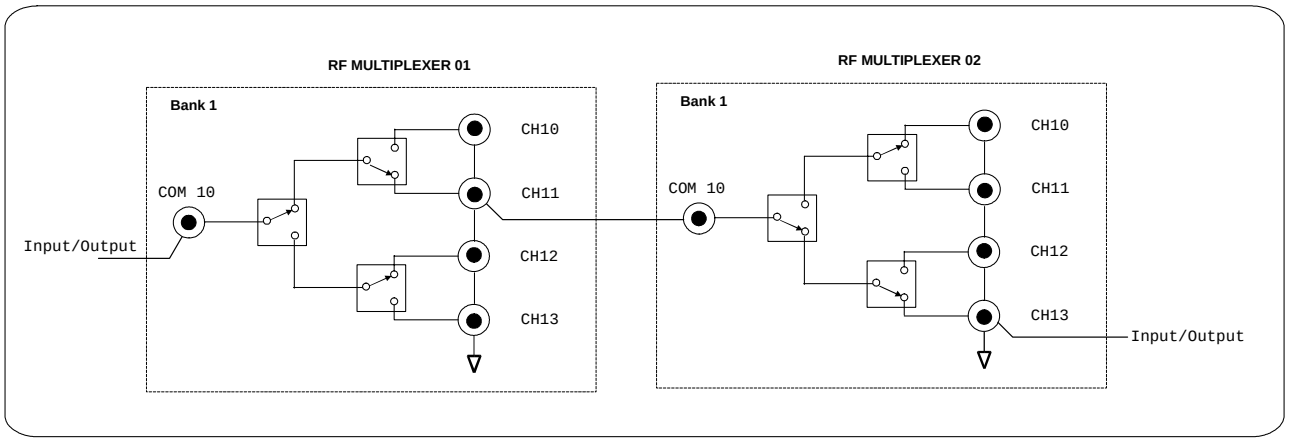


Figure 3-2. Example: RF Multiplexer Tree Switching

Example: Matrix-type Switching

Use matrix-type switching to provide connection of up to four devices under test (DUT) to up to four test instruments. With this configuration, only one channel in bank 0 (one "row") can be connected to one channel in bank 1 (one "column") at a time.

This example uses one RF Multiplexer and one Expander in a matrix-type configuration to connect channel 03 of the RF Multiplexer (module 00) to channel 11 of the Expander (module 01). The RF Multiplexer and Expander modules are defined as a single-multiplexer switchbox instrument. As shown in Figure 3-3, to connect channel 03 to channel 11, execute: CLOS (@10003,10111). The programs for this example can be written by referring to those provided for “Example: Standard Switching” on page 34.

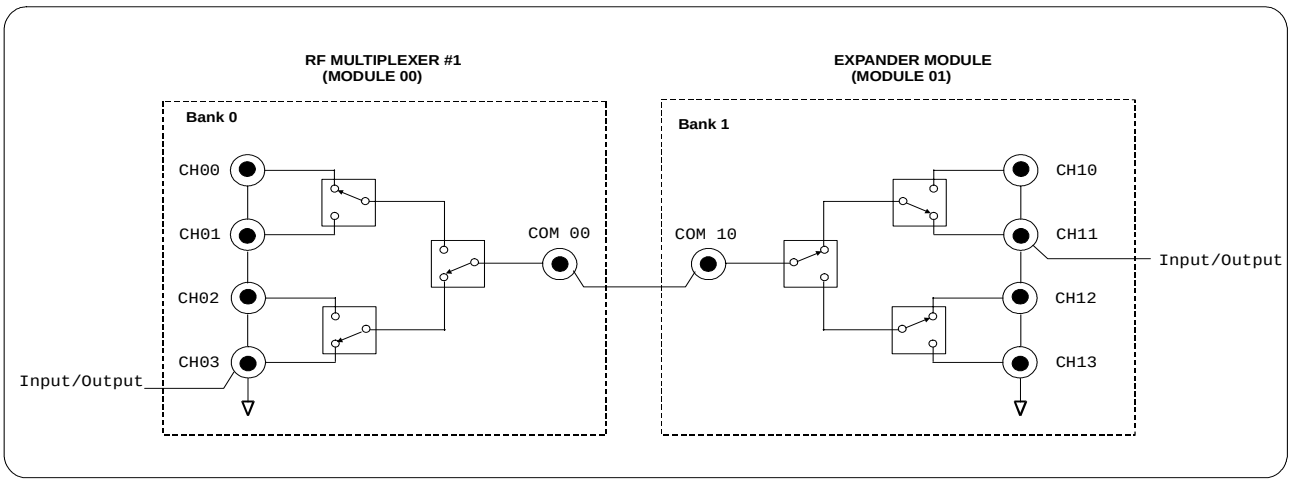


Figure 3-3. Example: RF Multiplexer/Expander Matrix-Type Switching

Recalling and Saving States

This section contains information about saving and recalling current RF Multiplexer states.

The *SAV <*numeric_state*> command saves the current channel states (which channel is being closed to its Common connector). The state number (0-9) is specified by the <*numeric_state*> parameter.

The *RCL <*numeric_state*> command recalls a previously saved state specified by the <*numeric_state*> parameter. If *SAV was not previously executed using the selected *numeric_state*, the RF multiplexer will be configured to the default reset values (channel *n*0 connected to COM *n*0 on all banks).

Example: Saving and Recalling Instrument State (HTBasic)

The following HTBasic program shows how to save and recall the RF Multiplexer channel states. Assuming the RF Multiplexer, without Expander modules connected, is defined as a single-multiplexer switchbox instrument. It first closes channels 101, 112, and 123, then saves current channel states to the state 5. After reset the module, then recall the stored state 5 and verify whether the channels are set to the saved state.

```
10  DIM A$[100]                          ! Dimension a string variables.
20  OUTPUT 70915; "*RST; *CLS"           ! Reset the module and clear its
                                           Status Register.

30  OUTPUT 70915; "CLOS (@101,112,123)"
                                         ! Close channel relays.
40  OUTPUT 70915; "*SAV 5"               ! Save all channel states as
                                           numeric state 5.

50  OUTPUT 70915; "*RST; *CLS"           ! Reset the module and clear
                                           Status Register
                                           .
60  OUTPUT 70915; "CLOS? (@101,112,123)"
                                         ! Query the channel states after
                                           a reset. "0,0.0" should be
                                           returned.
70  ENTER 70915;A$
80  PRINT "After reset, the channel states: ";A$
                                         ! Display the channel states.
90  OUTPUT 70915; "*RCL 5"               ! Recall the state 5.

100 OUTPUT 70915; "CLOS? (@101,112,123)"
                                         ! Query to see whether the
                                           channels are set to the saved
                                           state.

110 ENTER 70915;A$
120 PRINT "After recalling, the channel states: ";A$
                                         ! "1,1,1" should be returned.
130 END
```

Detecting Error Conditions

The SYSTem:ERRor? command queries the instrument's error queue for error conditions. If no error occurs, the RF Multiplexer module responds with 0,"No error". If errors do occur, the module will respond with the first one in its error queue. Subsequent queries continue to read the error queue until it is empty. The response takes the following form:

<*err_number*>, <*err_message*>

where <*err_number*> is an integer ranging from -32768 to 32767, and the <*err_message*> is a short description of the error and the maximum string length is 255 characters.

Example: Querying Errors (HTBasic)

The following example program was written in HTBasic programming language. It attempts an illegal channel closure for the E8482A RF Multiplexer module, then polls for the error message.

```
10  DIM Err_num$[256]                ! Dimension a string variable.
20  OUTPUT 70915; "CLOS (@160)"      ! Try to close an illegal channel.
30  OUTPUT 70915; ":SYST:ERR?"       ! Check for a system error.
40  ENTER 70915;Err_num$             ! Enter the error into Err_num$.
50  PRINT "Error: ";Err_num$         ! Print error +2001, "Invalid
                                       channel number".
60  END
```

Querying the RF Multiplexer

This section summarizes the query commands you can use to determine the configuration or state of the RF Multiplexer. All commands end with the "?" which puts the data into the output buffer where you can retrieve it to your computer. See Chapter 4 for more information on these commands.

Channel closed:	CLOS?
Channel open:	OPEN?
Module Description:	SYST:CDES?
Modules Installed:	SYST:COPT?
Module Type:	SYST:CTYP?
System error:	SYST:ERR?

Synchronizing the Instruments

This section discusses synchronizing the RF Multiplexer module with other instruments when making measurements.

In the following example, the RF Multiplexer switches a signal to be measured by a multimeter. This program verifies that the switching is complete before the multimeter begins a measurement.

Example: Synchronizing the Instruments (HTBasic)

This example program was written in HTBasic language. Assuming the multimeter (Agilent 34401A) has the GPIB address of 722 and the RF Multiplexer module has a logical address of 120 (GPIB address of 70915).

```
10   OUTPUT 70915; "*RST"                  ! Reset the module.
20   OUTPUT 70915; "CLOS (@101)"           ! Close channel 01.
30   OUTPUT 70915; "*OPC?"                 ! Wait for operation complete.
40   ENTER 70915; OPC_value
50   OUTPUT 70915; "CLOS? (@101)"          ! Verify that the channel is
                                             closed.
60   ENTER 70915; A
70   IF A=1 THEN
80      OUTPUT 722; "MEAS:VOLT:DC?"        ! When channel is closed, make
                                             the measure.
90      ENTER 722; Meas_value
100     PRINT Meas_value                   ! Print the measured value.
110  ELSE
120     PRINT "CHANNEL NOT CLOSE"
130  END IF
140  END
```

Chapter 4
Command Reference

About This Chapter

This chapter describes Standard Commands for Programmable Instruments (SCPI) and summarizes IEEE 488.2 Common (*) commands applicable to the RF Multiplexer. See the *Agilent E1406A Command Module User's Guide* for additional information on SCPI and common commands. This chapter contains the following sections:

Command Types

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

Common Command Format

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

```
*RST        *ESE 32        *STB?
```

SCPI Command Format

The SCPI commands perform functions like closing/opening switches, making measurements, 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>
```

[ROUTe:] is the root command, CLOSe is the second level sub command, and <*channel_list*> is a parameter.

Command Separator

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

```
ROUTe:CLOSe?
```

Colons separate the root command from the second level (ROUTe:CLOSe). If a third level existed, the second level is also separated from the third level by a colon.

Abbreviated Commands

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

For example, if the command syntax shows CLOSe, then CLOS and CLOSE are both acceptable forms. Other forms of CLOSe, such as CL or CLO will generate an error. You may use upper or lower case letters. Therefore, CLOSE, close, and ClOsE are all acceptable.

Implied Commands

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

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

The root command [ROUTe:] is an implied command. To make a query about a channel's present state, 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 parameter types you might see later in this chapter.

Parameter Type	Explanations and Examples
Numeric	Accepts all commonly used decimal representations of number including optional signs, decimal points, and scientific notation. 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01. Special cases include MINimum, MAXimum, and DEFault.
Boolean	Represents a single binary condition that is either true or false. ON, OFF, 1, 0
Discrete	Selects from a finite number of values. These parameters use mnemonics to represent each valid setting. An example is the TRIGger:SOURce <*source*> command where *source* can be BUS, EXT, HOLD, or IMM.

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

Linking Commands

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

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

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

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

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

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

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

SCPI Command Reference

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

The **DIAGnostic** subsystem is used to control the module's interrupt capability, including disabling the interrupt, selecting an interrupt line, as well as setting interrupt timer. All these settings can also be queried with this subsystem.

Subsystem Syntax

```
DIAGnostic
    :INTerrupt
        [:LINE]  <card_number>, <line_number>
        [:LINE]?  <card_number>
        :TIMer  <card_number>, <time_interval>
        :TIMer?  <card_number>
    :TEST
        [:RELays]?
        :SEEProm?  <card_number>
```

DIAGnostic:INTerrupt[:LINe]

DIAGnostic:INTerrupt[:LINe] <*card_number*>, <*line_number*> sets the interrupt line of the specified module. The <*card_number*> specifies which E8482A in a multiple-module switchbox is being referred to. The <*line_number*> can be 1 through 7 corresponding to VXI backplane interrupt lines 1 through 7.

NOTE *Changing the interrupt priority level is not recommended. DO NOT change it unless specially instructed to do so. Refer to the E1406A Command Module User's Manual for more details.*

Parameters

Name	Type	Range of Values	Default Value
<*card_number*>	numeric	1 - 99	N/A
<*line_number*>	numeric	0 - 7	1

Comments

Disable Interrupt: Setting <*line_number*> = 0 will disable the module's interrupt capability.

Select an Interrupt Line: The *line_number* can be 1 through 7 corresponding to VXI backplane interrupt lines 1 through 7. Only one value can be set at one time. The default value is 1 (lowest interrupt level).

Related Commands: DIAGnostic:INTerrupt:[LINe]?

Example

Setting Interrupt Line 1 for RF Multiplexer

```
DIAG:INT:LIN 1, 1                    ! Set the module's interrupt line to line 1.
```

DIAGnostic:INTerrupt[:LINe]?

DIAGnostic:INTerrupt[:LINe]? *<card_number>* queries the module's VXI backplane interrupt line and the returned value is one of 1, 2, 3, 4, 5, 6, 7 which corresponds to the module's interrupt lines 1-7. The returned value being 0 indicates that the module's interrupt is disabled. The *card_number* specifies which E8482A in a multiple-module switchbox is being referred to.

Parameters

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

Comments

Return value of "0" indicates that the module's interrupt is disabled. Return values of 1-7 correspond to VXI backplane interrupt lines 1 through 7.

When power-on or reset the module, the default interrupt line is 1.

Example

Querying the Module's Interrupt Line

```
DIAG:INT:LIN 1, 1          ! Set the interrupt line of Module #1 to
                             line 1.
DIAG:INT:LIN? 1            ! Query the module's interrupt line.
```

DIAGnostic:INTerrupt:TIMer

DIAGnostic:INTerrupt:TIMer *<card_number>, <timer>* is used to set the amount of time the module will wait after a relay close or open command is given before sending an interrupt and clearing the "busy" bit. The <*card_number*> parameter specifies which E8482A in a multiple-module switchbox is to be set.

Parameters

Name	Type	Range of Values	Default Value
<*card_number*>	numeric	1 - 99	N/A
<*timer*>	numeric	0.000001 - 0.064 seconds	0.01 second

Comments

We highly recommend you set the time to 10 ms for the E8482A relay.

*RST does not change the selected time.

NOTE *Setting the interrupt timer too small can cause system problems. We DO NOT recommend to change it unless specially instructed to do so.*

NOTE *The interrupt timer will default to 10 ms and can not be changed when the E8482A module is set to the E1472A Mode (see Page 26 of this manual for more detailed setting information).*

Example **Setting Interrupt Timer of Module #1 to 10 ms**

```
DIAG:INT:TIM 1, 0.01          ! Set interrupt timer on Module #1
                                to10 ms.
```

DIAGnostic:INTerrupt:TIMer?

DIAGnostic:INTerrupt:TIMer? <***card_number***> queries the specified module and returns the interrupt delay time set by the DIAG:INT:TIM command.

Example **Querying Module's Interrupt Timer**

```
DIAG:INT:TIM? 1               ! Query the interrupt timer setting for the
                                Module #1.
```

DIAGnostic:TEST[:RELays]?

DIAGnostic:TEST[:RELays]? causes the instrument to perform a self test which includes writing to and reading from all relay registers and verifying the correct values. A failure may indicate a potential hardware problem.

Comments **Returned Value:** Returns 0 if all tests passed; otherwise the card fails.

Error Codes: If the card fails, the returned value is in the form ***100*card number + error code***. Error codes are:

1 = Internal driver error;
2 = VXI bus time out;
3 = Card ID register incorrect;
5 = Card data register incorrect;
10 = Card did not interrupt;
11 = Card busy time incorrect;
40 = Relay register read and written data don't match.

WARNING **Disconnect any connections to the module when performing this function.**

Example **Performing Diagnostic Test to Check Error(s)**

```
DIAG:TEST?                    ! Returned value can be either 0 or other
                                value. "0" indicates that the system has
                                passed the self test otherwise the system
                                has an error.
```

DIAGnostic:TEST:SEEProm?

DIAGnostic:TEST:SEEProm? <*card_number*> checks the integrity (checksum) of the serial EEPROM on the module. Return value of "0" if no error. Otherwise, return value of "-1".

Parameters

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

Comments

Related Commands: SYST:CTYPE? <*card_number*>

Example

Checking EEPROM Checksum on Module #1

```
DIAG:TEST:SEEProm? 1                    ! Return "0" if no error.
```

The **DISPlay** subsystem monitors the channel state of the selected module in a switchbox. This subsystem operates with an E1406A command module when a display terminal is connected. With an RS-232 terminal connected to the E1406A command module's RS-232 port, these commands control the display on the terminal, and would in most cases be typed directly from the terminal keyboard. It is possible however, to send these commands over the GPIB interface, and control the terminal's display. In this case, care must be taken that the instrument receiving the DISPlay command is the same one that is currently selected on the terminal; otherwise, the GPIB command will have no visible affect.

Subsystem Syntax

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

DISPlay:MONitor:CARD

DISPlay:MONitor:CARD <***number***> **| AUTO** selects the module in a switchbox to be monitored when the monitor mode is enabled. Use the DISPlay:MONitor:STATe command to enable or disable the monitor mode.

Parameters

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

Comments

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

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

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

Example

Selecting Module #2 in a Switchbox for Monitoring

```
DISPlay:MONitor:CARD 2          ! Select module #2 in a switchbox to be monitored.
```

DISPlay:MONitor:CARD?

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

DISPlay:MONitor[:STATe]

DISPlay:MONitor[:STATe] <*mode*> turns the monitor mode ON or OFF. When monitor mode is on, the RS-232 terminal display the closed channel numbers of the module. The display is dynamically updated each time a channel is opened or closed.

Parameters

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

Comments

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

NOTE

Typing in another command on the RS-232 terminal will cause the DISPlay:MONitor[:STATe] *to automatically be set to OFF (0). Use of the OFF parameter is useful only if the command is issued over the GPIB interface.*

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

Monitor Mode for an E8482A: When monitoring mode is turned ON, all closed channels on the E8482A module (00-53) and the Expander modules #1 & #2 (100-153 & 200-253) will be displayed at the bottom of the terminal. The six decimal numbers within the two brackets correspond to the closed channel numbers of the six banks in the sequence of bank 0 to bank 5. For example, the display:

Chan: 00-53(102013) 100-153(010000) 200-253(000000)

The example indicates that channels 01, 10, 22, 30, 41 and 53 on the E8482A module, channels 00, 11, 20, 30, 40 and 50 on the Expander module #1, and channels 00, 10, 20, 30, 40 and 50 on the Expander module #2 are closed (connected to their Common connectors).

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

Example

Enabling the Monitor Mode for Module #2

```
DISP:MON:CARD 2          ! Select module #2 in a switchbox to be monitored.
DISP:MON ON              ! Turn on monitor mode.
```

DISPlay:MONitor[:STATe]?

DISPlay:MONitor[:STATe]? queries the monitor mode state to determine if it is set to ON or OFF.

The [ROUTe:] command subsystem controls switching operations for the RF Multiplexer modules in a switchbox.

Subsystem Syntax

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

[ROUTe:]CLOSe

[ROUTe:]CLOSe <*channel_list*> closes the RF Multiplexer channels specified by the *channel_list*. *Channel_list* is in the form of (@*ccmmnn*), where *cc* = card number (01-99), *mm* = module number (00-02), and *nn* = channel number (see table below).

Parameters

Name	Type	Range of Values	Items
channel_list	numeric	1 - 99	cc
	numeric	00 - 02	mm
	numeric	00-03, 10-13, 20-23, 30-33, 40-43, 50-53 (for E8482A) 00-03, 10-13, 20-23 (for E8482B)	nn

Comments

Channel Lists: When E1473A Expander(s) are NOT connected to an E8482A RF Multiplexer, <*channel_list*> has the form (@*ccnn*) (*mm* is omitted). When E1473A Expander(s) ARE connected to an E8482A RF Multiplexer, <*channel_list*> has the form (@*ccmmnn*) (*mm* MUST be included).

Closing Channels (no E1473A Attached to RF Multiplexer): To close a single channel, use [ROUTe:]CLOSe (@*ccnn*). To close multiple channels, use [ROUTe:]CLOSe (@*ccnn,ccnn,...*) (or any combination). Closure order for multiple channels with a single command is not guaranteed. For example, CLOSe (@101, 103, 105) closes channels 01, 03, and 05 on an RF Multiplexer when no E1473A Expanders are attached.

Closing Channels (E1473A(s) Attached to RF Multiplexer): To close a single channel, use [ROUTe:]CLOSe (@*ccmmnn*). To close multiple channels, use [ROUTe:]CLOSe (@*ccmmnn,ccmmnn,...*) (or any combination). Closure order for multiple channels with a single command is not guaranteed. For example, consider an RF Multiplexer with two E1473A Expander(s) attached (see Figure 2-2). Executing CLOSe (@10101, 10103, 10105) closes channels 01, 03, and 05 on E1473A Expander Module 01, but does not affect the RF Multiplexer (Module 00) or the E1473A Expander Module 02 channel states.

NOTE *Only one channel in each bank can be connected to its common at a time.*

Channel Range (no E1473A Attached to RF Multiplexer): The RF Multiplexer will accept and execute channel ranges without generating an error, but the result is to close the last channel in each bank within the range specified. For example, for an RF Multiplexer (card 1) without an E1473A Expander attached, after CLOSe (@101:151) is executed, channels 03, 13, 23, 33, 43, and 51 of the RF Multiplexer remain closed.

Channel Range (E1473A(s) Attached to RF Multiplexer): The RF Multiplexer will accept and execute channel ranges without generating an error, but the result is to close the last channel in each bank within the range specified. For example, consider an RF Multiplexer (card 1, module 00) with one E1473A Expander (card 1, module 01) attached (see Figure 2-2). When CLOSe (@10101:10151) is executed, the E1473A Expander (card 1, module 01) channels 03, 13, 23, 33, 43, and 51 remain closed. Since this CLOSe command affects only the E1473A Expander, channel states for the RF Multiplexer are not affected.

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

***RST Condition:** Channel n0 of all banks (n = 0-5 for E8482A or 0-2 for E8482B) are connected to their corresponding COM n0.

Example Closing RF Multiplexer Channels

This example closes channels 101 and 122 of a single-card (without Expander module) switchbox.

```
CLOS (@101,122)          ! Connect channel 01 to COM 00
                           and connects channel 22 to COM 20
                           on card #1.
```

[ROUTe:]CLOSe?

[ROUTe:]CLOSe? <*channel_list*> queries the current state of the specified channel(s). The *channel_list* is in the form of (@ccmmnn) (see [[ROUTe:]CLOSe command on page 50 for definition). The command returns a "1" if the channel is closed or a "0" if the channel is open. If a list of channels is queried, a comma delineated list of 1's and 0's is returned in the same order of the channel list.

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. A maximum of 127 channels at a time can be queried for a multi-module switchbox.

Example Querying Channel Closure State

This example closes channels for a single RF Multiplexer (card 01, module 00) with an E1473A Expander (card 01, module 01) attached. (See Figure 2-2 for a typical configuration.) Channels 10001 (channel 01 for the RF Multiplexer) and 10101 (channel 01 for the E1473A) are closed and the channels closure state is queried. Since the channels are programmed to be closed, "1,1" is returned.

```
CLOS (@10001,10101)          ! Connect channel 01 to COM00 on the
                               RF Multiplexer (card 01, module 00)
                               and channel 01 to COM00 on the
                               Expander (card 01, module 01).
CLOS? (@10001,10101)         ! Query channel closure state, returned
                               value "1,1" indicates that both channels
                               are closed.
```

[ROUTe:]OPEN?

[ROUTe:]OPEN? *<channel_list>* queries the current state of the specified channel(s). *Channel_list* is in the form of (@ccmmnn) (see [[ROUTe:]CLOSe command on page 50 for definition). The command returns a "1" if the channel is open or a "0" if the channel is closed. If a list of channels is queried, a comma delineated list of 1's and 0's is returned in the same order of the channel list.

Comments

Query is Software Readback: The ROUTe:OPEN? command returns the current software state of the channel(s) specified. It does not account for relay hardware failures. A maximum of 127 channels at a time can be queried for a multi-module switchbox.

Example

Querying Channel Open State

This example closes channels for a single RF Multiplexer (card 01, module 00) with an E1473A Expander (card 01, module 01) attached. (See Figure 2-2 for a typical configuration.) Channels 10001 (channel 01 for the RF Multiplexer) and 10101 (channel 01 for the E1473A) are closed and the channels closure state is queried. Since the channels are programmed to be closed, "0,0" is returned.

```
CLOS (@10001,10101)          ! Connect channel 01 to COM00 on the
                               RF Multiplexer module (00) and
                               channel 01 to COM00 on the Expander
                               module (01).
OPEN? (@10001,10101)         ! Query channel open states, returned
                               value "0,0" indicates that both channels
                               are closed.
```

The **SYSTem** subsystem returns the error numbers and error messages in the error queue of a switchbox, and returns the types of descriptions of cards and modules in a switchbox.

Subsystem Syntax

```
SYSTem
    :CDEScription? <card_number>
    :COPTion? <card_number>
    :CPON <card_number> | ALL
    :CTYPe? <card_number>
    :ERRor?
    :VERSion?
```

SYSTem:CDEScription?

SYSTem:CDEScription? *<card_number>* returns the description of a selected module in a switchbox.

Parameters

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

Comments

Module Description: The SYSTem:CDEScription? <*card_number*> command returns:

"Six/Three 1:4 RF Relay Multiplexers"

Example

Reading Module Description

```
SYST:CDES? 1                    ! Return the description of Card #1.
```

SYSTem:COPTion?

SYSTem:COPTion? *<card_number>* returns the model numbers of the RF Multiplexer module and the Expander Module(s) connected to it.

Parameters

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

Comments

The returned string contains three parts delineated by two commas. The first part, either E8482A or E8482B, indicates the model number of the RF Multiplexer (Module #00). The following two (E1473A or E1475A) are the model numbers of the Expander modules being connected (Module #01 and Module #02). If no Expander module is connected, 0 (zero) is in place of the model number.

For example, if "**`E8482A,E1473A,E1473A`**" is returned, it indicates the queried card is an E8482A RF Multiplexer module with two E1473A Expander modules connected. If "**`E8482B,E1473A,0`**" is returned, it indicates the queried card is an E8482B RF Multiplexer module with an E1473A connected to its RMD3/RMD2 internal ribbon connector and no Expander module is connected to its RMD5/RMD4 connector.

Example

Querying to Verify the Connected Expander Module(s)

```
SYST:COPT? 1                ! Return which expander module(s) are
                              connected. Returns"E8482A,0,0"
                              indicating no expander module is being
                              connected to the card 1.
```

SYSTem:CPON

SYSTem:CPON *<card_number> | ALL* resets the selected module, or multiple modules to their power-on state.

Parameter

Name	Type	Range of Values	Default Value
<card_number>	numeric	1 - 99	N/A

Comments

RF Multiplexer Module Power-On State: Channel n0 in all banks are connected to their corresponding COM n0 (n = 0-5 for E8482A or n = 0-2 for E8482B). Note that SYSTem:CPON ALL or *RST command connects channel n0 to COM n0 (n = 0-5) of all modules in a switchbox, while SYSTem:CPON <*card_number*> connects n0 to COM n0 (n = 0-5) in only the module (card) specified by the <*card_number*>.

Example

Setting Card #1 Module to Power-On State

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

SYSTem:CTYPe?

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

Parameters

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

Comments

RF Multiplexer Module Model Number: Sending this command returns:

```
HEWLETT-PACKARD,E8482A,<10-digit number>,A.11.01
```

where the <10-digit number> is the module's serial number (always 0) and A.11.01 is an example of the module revision code number.

NOTE *The <10-digit number> returns 0 (zero) if the checksum of the EEPROM on the module has error.The checksum of EEPROM on the module is always checked each time the* SYST:CTYP? <number> *command is executed. Refer to* DIAG:TEST:SEEProm? <card_*number*> *for details.*

Related Commands: DIAG:TEST:SEEProm? <*card_number*>

Example

Reading the Model Number of a Card #1

```
SYST:CTYP? 1                    ! Query the model number of Card #1.
```

SYSTem:ERRor?

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

Comments

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

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

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

***RST Condition:** *RST does not clear the error queue.

Example **Reading Error Queue**

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

SYSTem:VERSion?

SYSTem:VERSion? returns the version of the SCPI standard to which this instrument complies.

Comments **SCPI Version:** This command always returns a decimal value "1990.0", where "1990" is the year, and "0" is the revision number within that year.

Example **Reading SCPI Version**

```
SYST:VERS?                   ! Read the version of the SCPI standard.
```

SCPI Command Quick Reference

The following table summarizes the SCPI commands for the E8482A module.

Command		Description
DIAGnostic	:INTerrupt[:LINe] <*card_num*>,<*line_nu*m> :INTerrupt[:LINe]? <*card_num*> :INTerrupt:TIMer <*card_num*>,<*time*> :INTerrupt:TIMer? <*card_num*> :TEST[:RELays]? :TEST:EEPRom? <*card_num*>	Set an interrupt line for the specified module. Query the interrupt line of the specified module. Set wait time after an open or close before interrupting. Query the interrupt timer. Do diagnostic to find the specific error(s). Check the integrity (checksum) of EEPROM on the specified module.
DISPlay	:MONitor:CARD <*card_num*> | AUTO :MONitor:CARD? :MONitor[:STATe] <*mode*> :MONitor[:STATe]?	Select a module in a switchbox to be monitored. Query which module is set by above command. Set the monitor state on or off. Query the monitor state setting.
[ROUTe:]	CLOSe <*channel _list*> CLOSe? <*channel _list*> OPEN? <*channel _list*>	Close channel(s). Query channel(s) closed. Query channel(s) opened.
SYSTem	:CDEScription? <*card_num*> :COPTion? <*card_num*> :CPON <*card_num*> | ALL :CTYPe? <*card_num*> :ERRor? :VERSion?	Return description of the module. Return the RF Multiplexer/Expander modules configuration. Connect channel 0 to its COM 0 of all banks on the specified module(s). Return the module type. Return error number/message in the error queue. Return the version of the SCPI standard.

IEEE 488.2 Common Command Reference

The following table lists the IEEE 488.2 Common (*) Commands that can be accepted by the E8482A module.

Command	Command Description
*CLS	Clear Status Register: clears all Event Registers, the Request for OPC flag and all queues (except output queue).
*ESE <*mask*>	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 contents in the Standard Event Status Register.
*IDN?	Identification Query: returns identification string of the instrument.
*OPC	Operation Complete: sets the Request for OPC flag when all pending operations have completed. Also sets the 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.
*RCL <*state*>	Recall Saved State: recalls the instrument state previously saved by *SAV.
*RST	Reset the module: connects channel 0 to COM 0 on all banks.
*SAV <*state*>	Store the present instrument configuration in the specified memory location.
*SRE <*mask*>	Service Request Enable: used to set the Service Request Enable Register bits, and corresponding Serial Poll Status Byte Register bits, to generate a service request.
*SRE?	Service Request Enable Query: queries the current contents in the Service Request Enable Register.
*STB?	Status Byte Register Query: queries the current contents in the Status Byte Register.
*TST?	Self-test. Executes an internal self-test and returns only the first error encountered. Does not return multiple errors. The following is a list of responses you can obtain where "cc" is the card number with the leading zero deleted. +0 if self test passes. +cc01 for firmware error. +cc02 for bus error (problem communicating with the module). +cc03 for incorrect ID information read back from the module's ID register. +cc05 for hardware and firmware have different values. Possibly a hardware fault or an outside entity is register programming the E8482A. +cc10 if an interrupt was expected but not received. +cc11 if the busy bit was not held for a sufficient amount of time.
*WAI	Wait to Continue: halts execution of commands and queries until the "NO Operation Pending" message is true.

Appendix A
E8482A/B Specifications

Table 4-1. E8482A/B Specifications

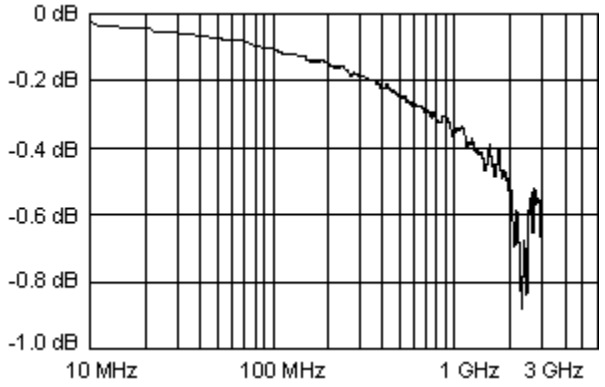
ITEMS		SPECIFICATIONS
GENERAL CHARACTERISTICS		
Module Size/Device Type:		C-Size 1-Slot, Register based, A16, slave only, P1 and P2 Connectors
Total Channels:		Six 1x4 Multiplexers (E8482A); Three 1x4 Multiplexers (E8482B)
Relays Type:		Non-latching armature
Typical Relay Life:	At rated load: (10 mA, 24 V, 2.5GHz, 50Ω impedance)	10^5
Power Requirements:	Peak module current: Dynamic module current:	0.1 A @ +5 V; 0.21 A @ +12 V 0.1 A @ +5 V; 0.1 A @ +12 V
Watts/slot:		6 W
Cooling/slot:		0.1 mm H_2O @ 0.5 Liter/sec for 10^{o}C rise
Operating Temperature:		0 - 55^{o}C
Operating Humidity:		65% RH, 0 - 40^{o}C
INPUT CHARACTERISTICS		
Maximum Voltage:	Any two terminals:	30 V dc, 30 V ac peak
Maximum Current:	Per channel or common:	0.5 A dc
Maximum Power:	Per channel or common:	10 Wdc, 10 VA ac
Characteristic Impedance:		50 Ω
DC ISOLATION		
Initial Closed Channel Resistance:		< 1 Ω
Isolation resistance: **(between any two terminals)**	< (40^{o}C, 65% RH):	> 10^6 Ω
Thermal Offset:	Per channel:	< 10 μV

(continued on the next page)

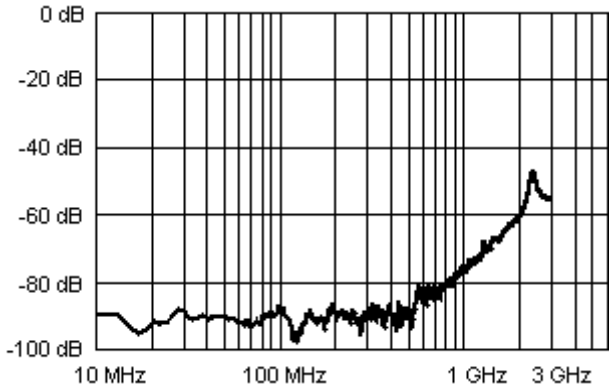
Table 4-1. E8482A/B Specifications (continued)

ITEMS		SPECIFICATIONS
AC ISOLATION / PERFORMANCE ($Z_I = Z_S = 50$ Ω, < (40°C, 95% RH):)		
Bandwidth (-1.5 dB):		2.5 GHz
Insertion Loss:	< 500 MHz: < 1 GHz: < 2 GHz: < 2.5 GHz: < 3 GHz:	< -0.5 dB < -0.6 dB < -1.0 dB < -1.5 dB < -2.0 dB (typical)
Crosstalk: (Closed channel to Closed Channel, with 50Ω termination)	< 500 MHz: < 1 GHz: < 2 GHz: < 2.5 GHz: < 3 GHz:	< -60 dB < -57 dB < -55 dB < -40 dB < -39 dB (typical)
VSWR (with 50 Ω termination)	< 500 MHz: < 1 GHz: < 2 GHz: < 2.5 GHz: < 3 GHz:	< 1.15 < 1.25 < 1.35 < 1.45 < 1.50 (typical)

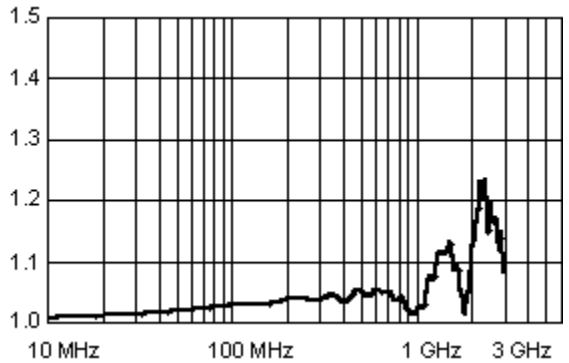
Typical Insertion Loss



Typical Crosstalk (Channel-to-channel)



Typical VSWR



Appendix B
Register-Based Programming

About This Appendix

The Agilent E8482A RF Multiplexer module is a register-based product which does not support the VXIbus word serial protocol. When a SCPI command is sent to the module, the instrument driver resident in the Agilent E1406A Command module parses the command and programs the module at the register level.

Register-based programming is a series of reads and writes directly to the module registers. This increases throughput speed since it eliminates the command parsing and allows the use of an embedded controller. Also, register programming provides an avenue for users to control a VXI module with an alternate VXI controller device and eliminate the need for using an E1406A Command module.

This appendix contains the information you need for register-based programming. The contents include:

Register Addressing

Register addresses for register-based devices are located in the upper 25% of VXI A16 address space. Every VXI device (up to 256 devices) is allocated a 32 word (64 byte) block of addresses. Figure B-1 on Page 62 shows the register address location within A16 as it might be mapped by an embedded controller. Figure B-2 on Page 63 shows the location of A16 address space in the Agilent E1406A Command module.

When you are reading from or writing to a register of the module, 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

Base Address

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

A16 Address Space Outside the Command Module

When the E1406A Command module is not part of your VXIbus system (Figure B-1), the module's base address is computed as:[1]

$C000_h + (LADDR_h * 40_h)$

***- or** (decimal)*

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

where $C000_h$ (49,152) is the starting location of the VXI A16 addresses, LADDR is the multiplexer's logical address, and 64 (40_h) is the number of address bytes per register-based module. For example, the module's logical address factory setting is 120 (78_h). If this address is not changed, the module will have a base address of:

$C000_h + (78_h * 40_h) = C000_h + 1E00_h = DE00_h$

***- or** (decimal)*

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

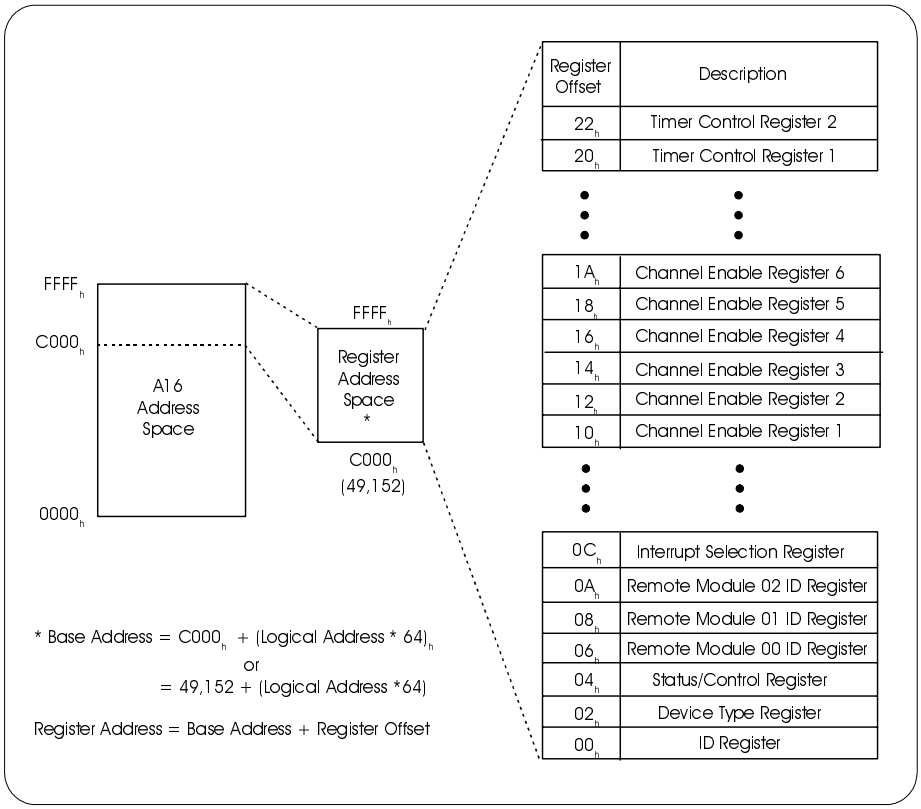


Figure B-1. Registers within A16 Address Space

1. Numbers with a subscripted "h" are in hexadecimal format. Numbers without the subscripted "h" are in decimal format.

A16 Address Space Inside the Command Module or Mainframe

When the A16 address space is inside the Agilent E1406A Command module (Figure B-2), the module's base address is computed as:[1]

$1FC000_h + (LADDR_h * 40_h)$

***- or** (decimal)*

2,080,768 + (LADDR * 64)

where $1FC000_h$ (2,080,768) is the starting location of the VXI A16 addresses, LADDR is the multiplexer's logical address, and 64 (40_h) is the number of address bytes per register-based device. Again, the multiplexer's factory setting logical address is 120 (78_h). If this address is not changed, the module will have a base address of:

$1FC000_h + (78_h * 40_h) = 1FC000_h + 1E00_h = 1FDE00_h$

***- or** (decimal)*

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

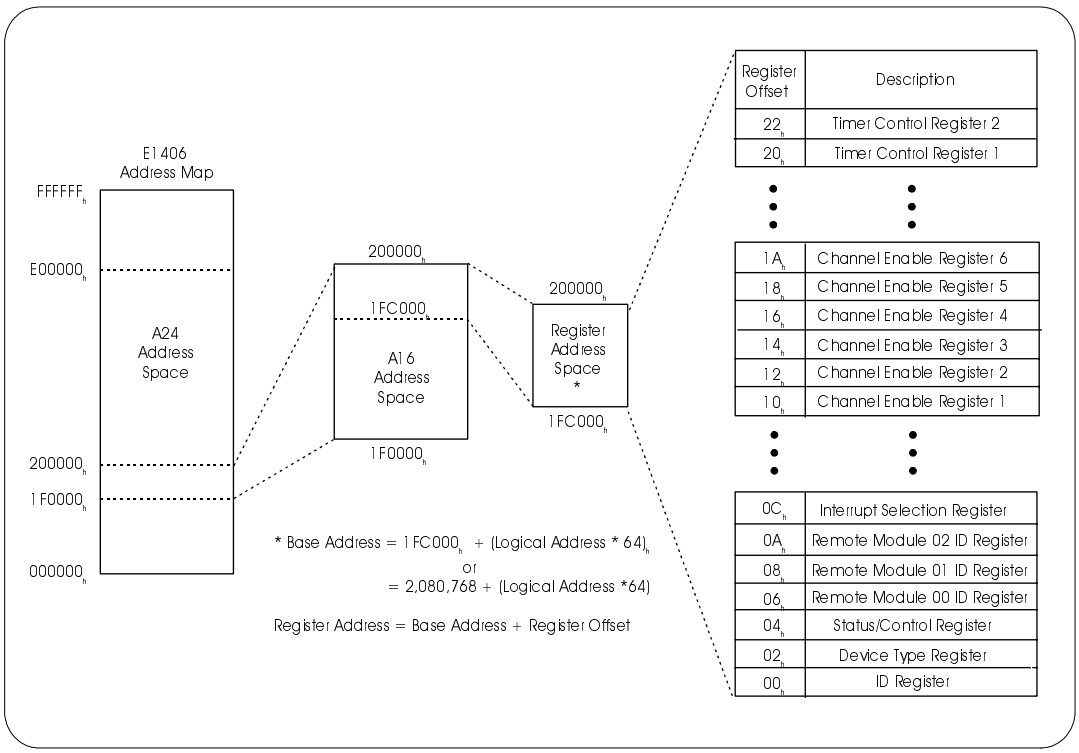


Figure B-2. Registers within Agilent E1406A A16 Address Space

1. Numbers with a subscripted "h" are in hexadecimal format. Numbers without the subscripted "h" are in decimal format.

Register Offset

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

$DE00_h + 04_h = DE04_h$
$1FDE00_h + 04_h = 1FDE04_h$

***- or** (decimal)*

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

Register Descriptions

The E8482A module contains 15 registers as shown in Table B-1. You can write to the writable (W) registers and read from the readable (R) registers. This section contains a description of the registers followed by a bit map of the registers in sequential address order.

NOTE *Undefined register bits (shown as "x" in the Tables) return as "1" when the register is read, and have no effect when written to.*

Table B-1. Module Registers

Registers	Addr. Offset	R/W	Register Address
Manufacturer ID Register	00_h	R	base + 00_h
Device Type Register	02_h	R	base + 02_h
Status/Control Register	04_h	R/W	base + 04_h
Remote Module 00 ID Register	06_h	R	base + 06_h
Remote Module 01 ID Register	08_h	R	base + 08_h
Remote Module 02 ID Register	$0A_h$	R	base + $0A_h$
Interrupt Selection Register [a]	$0C_h$	R/W	base + $0C_h$
Module 00 Bank 0-2 Channel Enable Register	10_h	R/W	base + 10_h
Module 00 Bank 3-5 Channel Enable Register	12_h	R/W	base + 12_h
Module 01 Bank 0-2 Channel Enable Register	14_h	R/W	base + 14_h
Module 01 Bank 3-5 Channel Enable Register	16_h	R/W	base + 16_h
Module 02 Bank 0-2 Channel Enable Register	18_h	R/W	base + 18_h
Module 02 Bank 3-5 Channel Enable Register	$1A_h$	R/W	base + $1A_h$
Timer Control Register 1[a]	20_h	R/W	base + 20_h
Timer Control Register 2[a]	22_h	R/W	base + 22_h

a. These registers are not effective when the E8482A module is set to the E1472A mode (as shown on Page 26).

ID Register

Reading the ID register (base + 00_h) returns $FFFF_h$ indicating the manufacturer is Agilent Technologies and the module is an A16 register-based device.

base + 00_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	x															
Read	Manufacturer ID - returns $FFFF_h$ indicating Agilent A16 only register-based card															

Device Type Register

Reading the Device Type Register (base + 02_h) returns $02D2_h$ which identifies the device as the E8482A/B RF Multiplexer module, with or without Expander module(s) connected. However, the relay module will generate a different remote module code to distinguish between the modules (see Remote Module ID Registers on Page 67).

base + 02_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	x															
Read [a]	$02D2_h$															

a. When the module is configured to be in E1472A Mode (as shown on Page 26), "0180_h" is returned.

Status/Control Register

The Status/Control Register is at offset address 04_h. It is used to control the module and inform the user of its status.

base + 04_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write [a]	x									IRQ E/D	x				S	Reset
Read [b]	x	MS	x						B	IRQ E/D	x		1	P	x	

a. Writing to the reserved bits ("x") will cause no action. We recommend writing "1" to these bits.
b. Reading from the reserved bits ("x") will return as "1". Do not rely on these values for card operation.

Reading the Status/Control Register

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

- **Self-test Passed (bit 2)** - Used to inform the user of the self-test status. "1" in this field indicates the module has successfully passed its self-test, and "0" indicates that the module is either executing or has failed its self-test.

- **Interrupt Status (bit 6)** - Used to inform the user of the interrupt status. "0" indicates that the interrupt is enabled, and "1" indicates that the interrupt is disabled. The interrupt generated after a channel has been closed can be disabled.

- **Busy (bit 7)** - Used to inform the user of a busy condition. "0" indicates that the module is busy, and "1" indicates that the module is not busy. Each relay requires about 10 ms execution time during which time the module is busy.

- **Modid Select (bit 14)** - "0" in this bit indicates that the module is selected by a high state on the P2 MODID line, and "1" indicates it is not selected via the P2 MODID line.

As an example, if a read of the Status Register (base + 04_h) returns "FFBF (1111111110111111)", it indicates that the module is not busy (bit 7 = 1) and the interrupt is enabled (bit 6 = 0).

Writing to the Status/Control Register

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

- **Soft Reset (bit 0)** - Writing a "1" to this bit will force the module to reset (channel 0 connected to its COM 0 on all banks).

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

- **Sysfail Inhibit (bit 1)** - Writing a "1" to this bit will disable the module from driving the SYSFAIL line (all channels open). The Slot-0 module can detect the failed module via this line.

- **Interrupt Enable/Disable (bit 6)** - Writing a "1" to this bit will disable the module from sending an interrupt request generated when channel relays are closed. Writing a "0" to this bit will enable the module's interrupt capability.

NOTE *Typically, interrupts are only disabled to "peek-poke" a module. Refer to your command module's operating manual before disabling the interrupt.*

Remote Module ID Registers

There are three Remote Module ID registers (base + 06_h - $0A_h$) used to help you to identify the number and location of the Expander modules (E1473A and/or E1475A) connected to the E8482A/B RF Multiplexer module. They are:

-- Remote Module 00 ID Register (at offset address 06_h)
-- Remote Module 01 ID Register (at offset address 08_h)
-- Remote Module 02 ID Register (at offset address $0A_h$)

The Remote Module ID Registers bit definitions are listed as below:

Remote Module 00 ID Register (base + 06_h)

base + 06_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	Module 00 ID															

Remote Module 01 ID Register (base + 08_h)

base + 08_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	Module 01 ID															

Remote Module 02 ID Register (base + $0A_h$)

base + $0A_h$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	Module 02 ID															

All these registers are read-only registers. When reading the three Remote Module ID Registers,

- The base $+06_h$ register returns module 00 status (RF Multiplexer relays connected to the RMD0/1 connector):

  -- Returns $FF22_h$ when E8482A RF Multiplexer relays are connected.
  -- Returns $FFF2_h$ when E8482B Multiplexer relays are connected.

- The base $+08_h$ register returns module 01 status (Expander module connected to the RMD2/3 connector).

  -- Returns $FFFF_h$ when Expander module number 01 is not connected.
  -- Returns $FF00_h$ when E1473A Expander module number 01 is connected.
  -- Returns $FF11_h$ when E1475A Expander module number 01 is connected.

- The base $+0A_h$ register returns module 02 status (Expander module connected to the RMD4/5 connector).

  -- Returns $FFFF_h$ when Expander module number 02 is not connected.
  -- Returns $FF00_h$ when E1473A Expander module number 02 is connected.
  -- Returns $FF11_h$ when E1475A Expander module number 02 is connected.

Interrupt Selection Register

The Interrupt Selection Register is at offset address $0C_h$. It is used to set the interrupt level of the module and inform the user of the current interrupt level being set.

base + $0C_h$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	x													Interrupt Level		
Read	x													Interrupt Level		

- You can set the interrupt level of the module by writing to **Interrupt Level Bits (bits 0-2)** of the register. Writing bits 2-0 with 001, 010, 011, 100, 101, 110, or 111 will set the interrupt level to 1 through 7 which corresponds to the VXI backplane lines 1-7. The highest interrupt level is 7, and the lowest level is 1 (default value).
- Reading the register will return the current interrupt level of the module. The returned value 001, 010, 011, 100, 101, 110, or 111 in Bits 2-0 corresponds to interrupt level 1 through 7.

NOTE *If the E8482A module is set to the E1472A mode, the Interrupt Selection Register is invalid. See Page 26 of this manual for more detailed setting information.*

Channel Enable Registers

There are six Channel Enable registers (base + 10_h - $1A_h$) driving the channel relays of the Multiplexer and the Expander Modules. All these registers are readable/writable (R/W) registers. Writing to the Channel Enable Registers (base + 10_h - $1A_h$) allows you to close desired channel to its COMmon. The data can be readback via the same address that the data being written to.

The bit definitions of these registers are listed as below:

Module 00 Bank 0 - 2 Channel Enable Register - RMD0 (base + 10_h)

base + 10_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined				ch23	ch22 ch23	ch21	ch20	ch13	ch12 ch13	ch11	ch10	ch03	ch02 ch03	ch01	ch00
Read																

Module 00 Bank 3 - 5 Channel Enable Register - RMD1 (base + 12_h)

base + 12_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined				ch53	ch52 ch53	ch51	ch50	ch43	ch42 ch43	ch41	ch40	ch33	ch32 ch33	ch31	ch30
Read																

Module 01 Bank 0 - 2 Channel Enable Register - RMD2 (base + 14_h)

base + 14_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined				ch23	ch22 ch23	ch21	ch20	ch13	ch12 ch13	ch11	ch10	ch03	ch02 ch03	ch01	ch00
Read																

Module 01 Bank 3 - 5 Channel Enable Register - RMD3 (base + 16_h)

base + 16_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined				ch53	ch52 ch53	ch51	ch50	ch43	ch42 ch43	ch41	ch40	ch33	ch32 ch33	ch31	ch30
Read																

Module 02 Bank 0 - 2 Channel Enable Register - RMD4 (base + 18_h)

base + 18_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined				ch23	ch22 ch23	ch21	ch20	ch13	ch12 ch13	ch11	ch10	ch03	ch02 ch03	ch01	ch00
Read																

Module 02 Bank 3 - 5 Channel Enable Register - RMD5 (base + $1A_h$)

base + $1A_h$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined				ch53	ch52 ch53	ch51	ch50	ch43	ch42 ch43	ch41	ch40	ch33	ch32 ch33	ch31	ch30
Read																

- Writing "1" to the bit(s) will close the related channel to its COMmon. Only one channel per bank can be closed at one time. All other bits in the bank are zeroes.

- Reading the Channel Enable Registers returns a hexadecimal number indicating the state of the relay driver circuit. It cannot detect a defective relay. Do not rely on the contents of the registers for normal operation.

For example, write "1" to bits 3 and 2 of the Module 00 Bank 3-5 Channel Enable register (base + 12h) to close channel 33 on the RF Multiplexer Module. Bits 0 and 1 of this register must be set to "0", since only one channel per bank can be closed at a time.

Timer Control Registers

Each relay on the E8482A module requires about 10 ms settling time during which time the module is busy. There are two registers used to provide a programmable timer for the relay settling time. They are:

-- Timer Control Register 1 (base + 20_h)

-- Timer Control Register 2 (base + 22_h)

The Timer Control Registers bit definitions are listed as below:

Timer Control Register 1 (base + 20_h)

base + 20_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Set Time															
Read	Read Time															

Timer Control Register 2 (base + 22_h)

base + 22_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	x								Set Time							
Read									Read Time							

As shown in above table, totally 24-bits of the two Timer Control Registers can be used to preset the relay settling time. Since the system clock is 16 MHz, each step of the timer is 62.5 nanoseconds. The maximum programmable timer can be set to:

$$62.5 \text{ ns} * \text{FFFFFF}_h = 1.0486 \text{ second}$$

The settling time is calculated based on the following formula:

$$\text{Settling Time} = 62.5 \text{ ns} * (\text{FFFFFF}_h - xxyyyy_h)$$

where $yyyy_h$ is the value written to the Timer Control Register 1 and xx_h to the Timer Control Register 2.

For example, if you want to preset the relay settling time to 10 ms, you should write 8EFF_h to Timer Control Register 1 (base + 20) and FD_h to Timer Control Register 2 (base + 22). Since:

$$xxyyyy_h = \text{FFFFFF}_h - (10 \text{ ms} / 62.5 \text{ ns})_h = \text{FD8EFF}_h$$

These two registers can also be read back. The returned value can be used to calculate the settling time being set for the relays.

NOTE *For register-based programming, you must set the relay settling time each time the module is powered up. We highly recommend you set the time to 10 ms for the E8482A relays.*

NOTE *If the E8482A module is set to the E1472A Mode (see Page 26 of this manual for more detailed setting information), the Timer Selection Registers are invalid.*

Notes:

Appendix C
Error Messages

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

Table C-1. RF Multiplexer Error Messages

Number	Error Message	Potential Cause(s)
-224	Illegal parameter	Attempting to execute a command with a parameter not applicable to the command.
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 module in a switchbox that is not supported by the module (e.g., channel 99 of a multiplexer module).
2006	Command not supported on this card	Sending a command to a module (card) in a switchbox that is unsupported by the module.
2009	Too many channels in channel list	Attempting to address more channels than available in the switchbox.
2010	Scan mode not supported on this card	Sending a command to a module (card) in a switchbox that is unsupported by the module.
2600	Function not supported on this card	Sending a command to a module (card) in a switchbox that is not supported by the module or switchbox.
2601	Channel list required	Sending a command requiring a channel list without the channel list.

Notes:

Index

M (*continued*)

N

O

P

Q

R

S

S (*continued*)

T

W

Agilent Technologies

Manual Part Number: E8482-90001
Printed in U.S.A. E0301