Agilent Technologies E8483A Microwave Switch/ Attenuator Driver Module User's Manual

Agilent Technologies

Manual Part Number: E8483-90001
Printed in U.S.A. E0301

Contents

Agilent E8483A User's Manual

AGILENT TECHNOLOGIES WARRANTY STATEMENT

AGILENT PRODUCT: E8483A Microwave Switch/Attenuator Driver 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

E8483A Microwave Switch/Attenuator Driver Module User's Manual
Edition 1

Documentation History

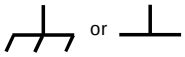
All Editions and Updates of this manual and their creation date are listed below. The first Edition of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct or add additional information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

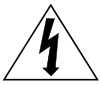
Indicates the field wiring terminal that must be connected to earth ground before operating the equipment — protects against electrical shock in case of fault.



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: Microwave Switch/Attenuator Driver Module
Model Number: E8483A
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 E8483A Microwave Switch/Attenuator Driver card, contains information on how to program it using SCPI (Standard Commands for Programmable Instruments) commands, and provides an example program to check initial operation. Chapter contents are:

Microwave Switch/Attenuator Driver Card Description

The Agilent E8483A is a VXIbus C-Size register-based product which can operate in a C-Size VXIbus mainframe. It can be used for driving up to six microwave switches and/or attenuators which are not supplied with the module. Whichever switches or attenuators you choose must be ordered separately. Because of the size of the microwave switches to be installed, the module takes up two C-size slots.

As shown in Figure 1-1, the mounting holes (labeled with Switch 0, Switch 1 and Switch 2) on the module's front panel allows the user to install up to three microwave switches on the module. The six 16-pin ribbon cable connectors (Groups 0-5) on the PC board of the module accommodate the need for connecting microwave switches, and the six 10-pin ribbon cable connectors (Groups 0-5) for connecting step attenuators. Totally, six switches and/or attenuators can be connected to, and controlled by the module at a time. In addition, six ribbon cables for switches and six ribbon cables for attenuators are provided along with the module, which makes the connection more conveniently. See Page 28 of this manual for more connecting information.

NOTE *DO NOT connect a microwave switch and an attenuator to the connectors within the same group. That is, if a microwave switch is connected to the 16-pin connector (i.e. Group 0), then you can not connect an attenuator to the 10-pin connector of the same group (e.g., Group 0). Otherwise, there may cause damage to the module. See "Guidelines for Connecting Switches/Attenuators" on page 28 of this manual for details.*

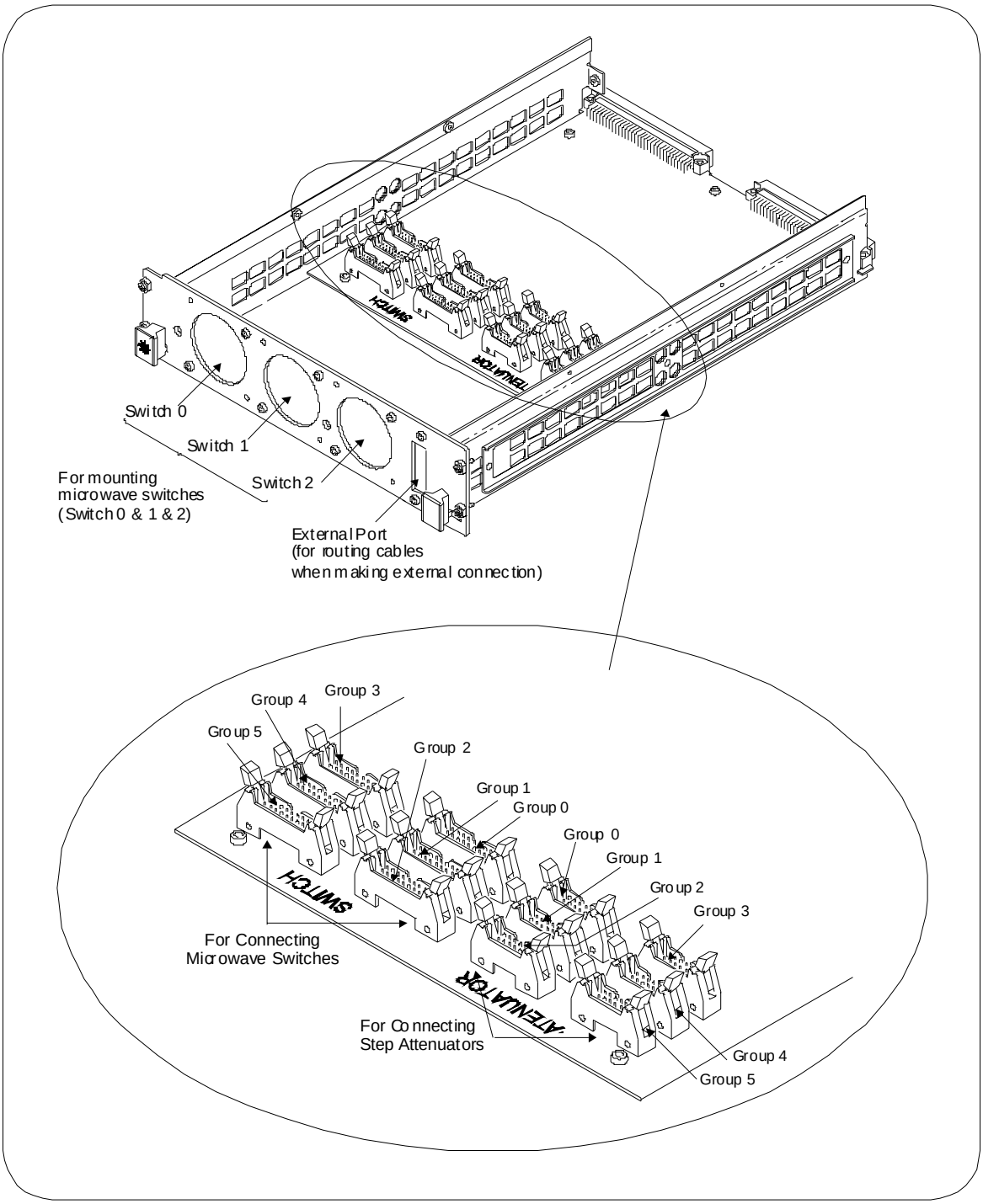


Figure 1-1. Agilent E8483A Module

Simplified Schematic of E8483A Driver Card

The E8483A driver board contains the control, drive and power circuitry for controlling up to six multiport microwave switches or step attenuators which are not supplied with the module. The power circuitry (with short circuit protection) can simultaneously provide 1.25 A at 24 Volts to all contacts for control of the switches and attenuators, so no external power supply is needed. Figure 1-2 shows the simplified schematic of the module for driving one multiport switch or one step attenuator. See Page 27 of this manual for the pinout information of the 16-pin connectors for connecting switches and 10-pin connectors for connecting attenuators.

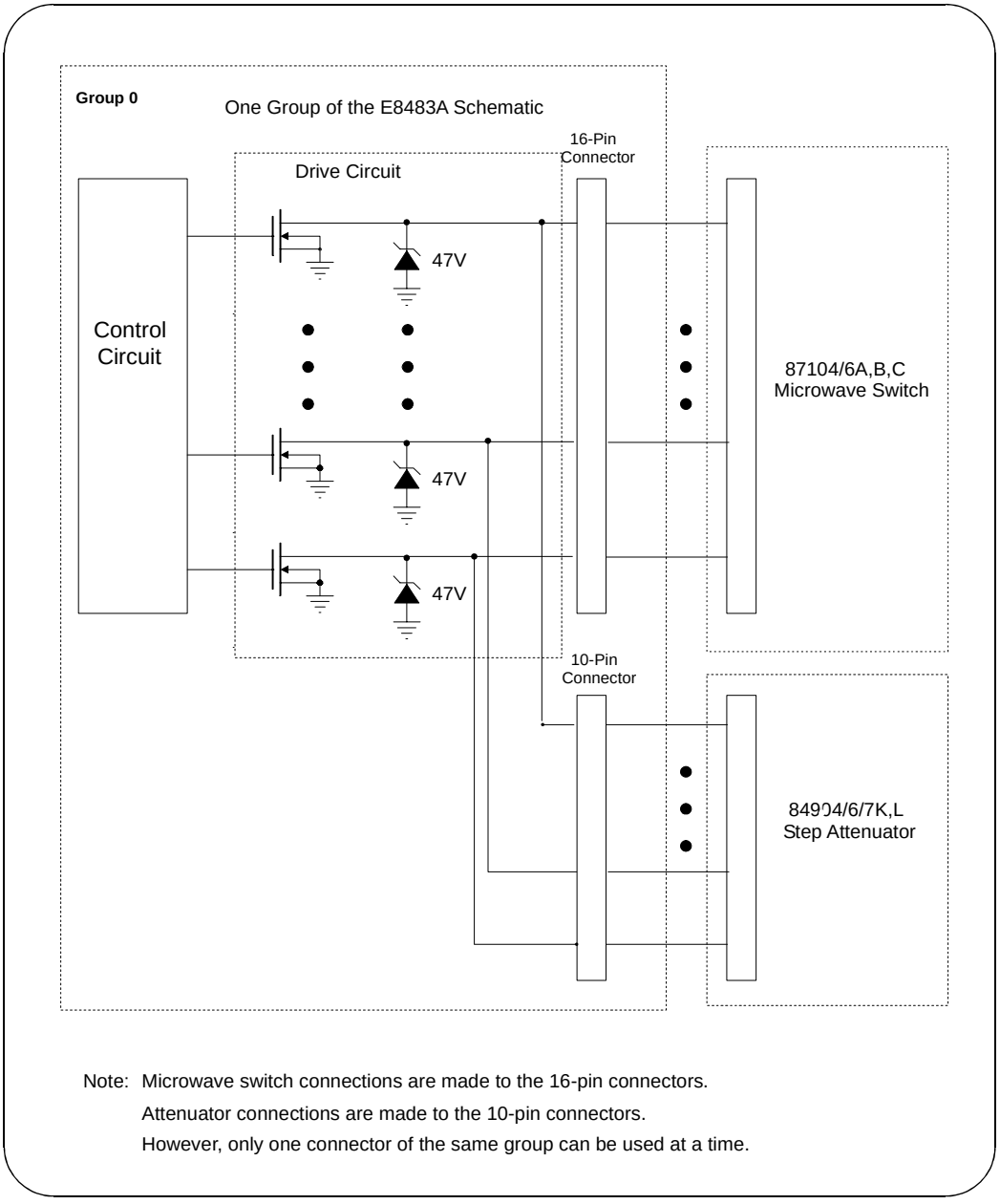


Figure 1-2. Simplified Schematic of the E8483A

Simplified Schematic of Microwave Switches

All Agilent 87104A/B/C and 87106A/B/C series microwave switches can be controlled by the E8483A Driver Card. Figure 1-3 shows the internal switch diagram for the various multiport microwave switches. At power-up, all paths of the microwave switches are open. During switching, these switches have an internal circuit that provides logic to open all but the selected ports, and then closes the selected paths. All other paths are terminated with 50 Ω loads, and the current to all of the solenoids is then cut off. These microwave switches also offer independent indicators that are controlled by the switch. The indicators provide a closed path between the indicator common pin and the corresponding sense pin of the selected path. For more information about these microwave switches, see the corresponding Technical Data Sheets.

NOTE *For these microwave switches, keep the switching time at the minimum of 15 ms to make sure that the switch is fully latched.*

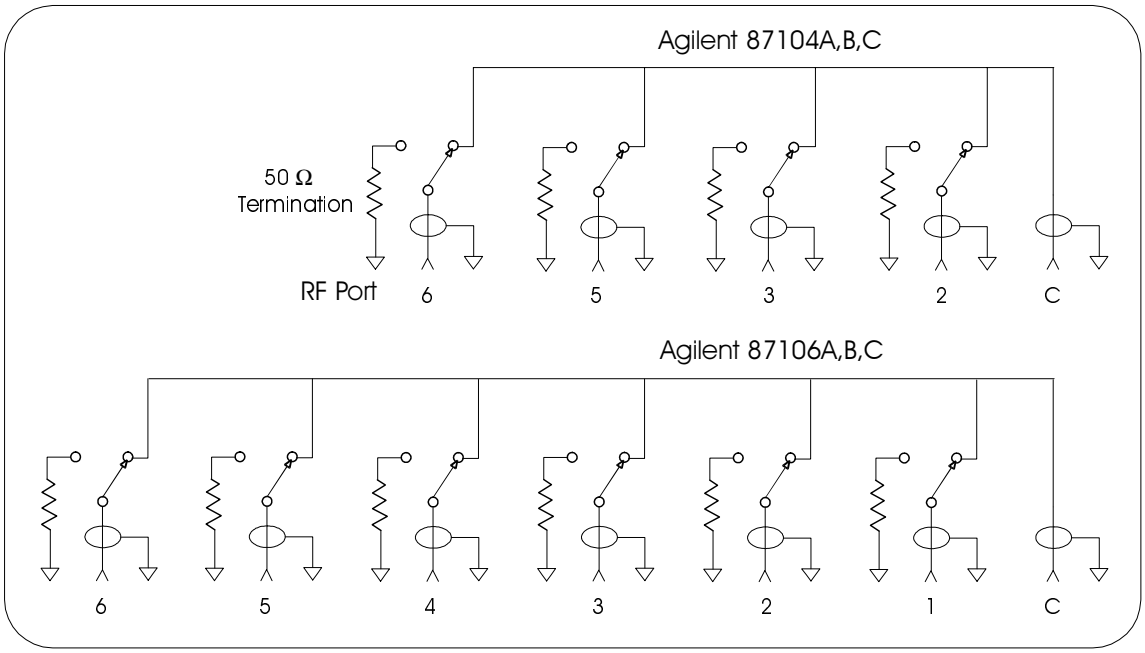


Figure 1-3. Multiport Microwave Switch Diagram

Simplified Schematic of Attenuators

All Agilent 84904K/L, 84906K/L, and 84907K/L series step attenuators can be controlled by the E8483A Driver Card. Figure 1-4 shows one attenuator section schematic. Each section utilizes one solenoid with dual coil windings, one coil to switch in the attenuator card (i.e. 10dB) and one coil to switch in the Thru line (0 dB). With positive voltage applied to the common pin, the state (attenuator card or thru line) of a particular section is determined by connecting its attenuator card pin or thru pin to a negative voltage or ground. As a section is switched, the internal contacts of the activated coil open, thus shutting off current flow. At the same time, the internal contacts for the other coil close so that it can be activated when desired. For more information about these attenuators, see the corresponding Technical Data Sheets.

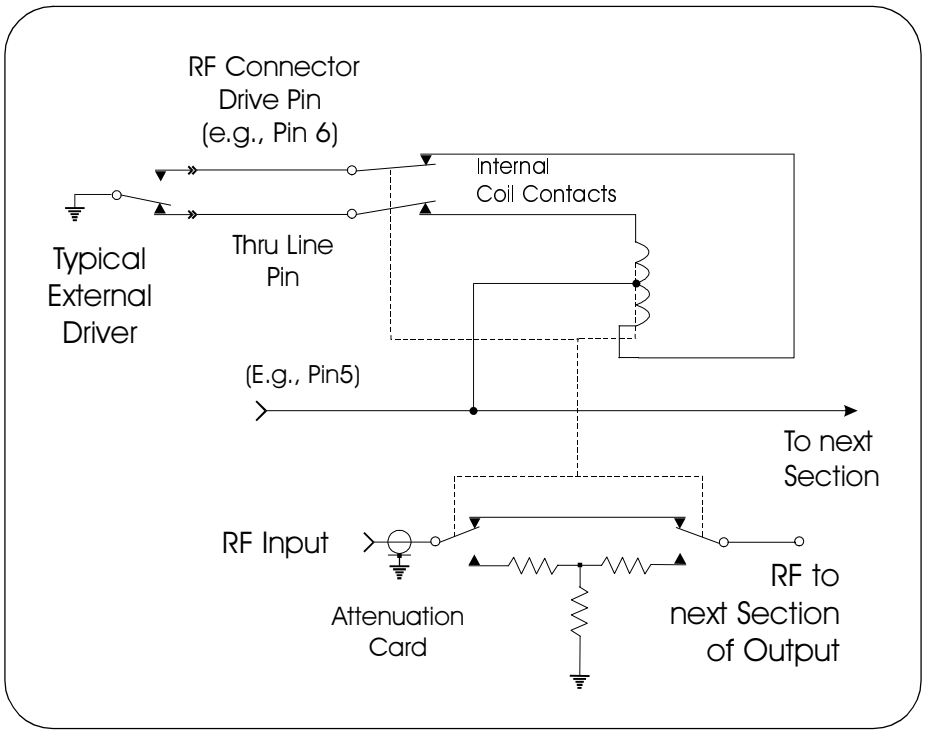


Figure 1-4. One Section Schematic of an Attenuator

NOTE *Although all sections can be switched simultaneously, the attenuator driver must not allow both pins of the same section to be activated concurrently, or else that section would cycle rapidly. Switching time is a maximum of 20 ms, including contact settling time.*

Typical Configuration

For a Standard Commands for Programmable Instruments (SCPI) environment, one or more E8483A modules can be configured as a switchbox instrument. For a switchbox instrument, all modules within the instrument can be addressed using a single interface address.

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 Module

To program the module using SCPI commands, you must select the controller language, interface address, and appropriate commands. See the *C-Size VXIbus System Installation and Getting Started 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 *The module can also be programmed by writing directly to its registers. See Appendix B for details on register programming.*

As shown in the following sections, both microwave switches and attenuators can be controlled by programming the E8483A module with SCPI commands.

Programming with Microwave Switches

To control microwave switches with the E8483A Driver module, You must specify the appropriate SCPI command and channel addresses. Table 1-1 lists the most commonly used commands to control switches. Refer to Chapter 4 of this manual for details of these commands.

Table 1-1. Commonly Used SCPI Commands for Switches

SCPI Commands	Commands Description
CLOSe <*channel_list*>	Connects the specified channel of the microwave switch to its common (C) port.
OPEN <*channel_list*>	Disconnect the specified port of the switch from its common (C) port..
SCAN <*channel_list*>	Closes the set of relays, one at a time.

Channel Addresses

When the E8483A is used to control switches, only valid channel addresses can be included in a *channel_list.* The channel address has the form of (@*ccgs*) where

cc = card number (01-99)
gs = channel number (switch type dependent, see Table 1-2)

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

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

NOTE *Only valid channels can be accessed in a channel_list or channel range. Also, the channel range must be from a lower channel number to a higher channel number. Otherwise, an error will be generated.*

Card Number

The card number (*cc* of the *channel_list*) identifies the E8483A module within a switchbox. The card number assigned depends on the switchbox configuration used. Leading zeroes can be ignored for the card number.

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

- **Multiple-module Switchbox**. In a multiple-module switchbox configuration, modules are set to successive logical addresses. The module with the lowest logical address is always card number 01. The module with the next successive logical address is card number 02, and so on. Figure 1-5 illustrates the card numbers and logical addresses of a typical multiple-module switchbox installed in an Agilent C-Size mainframe with an Agilent command module.

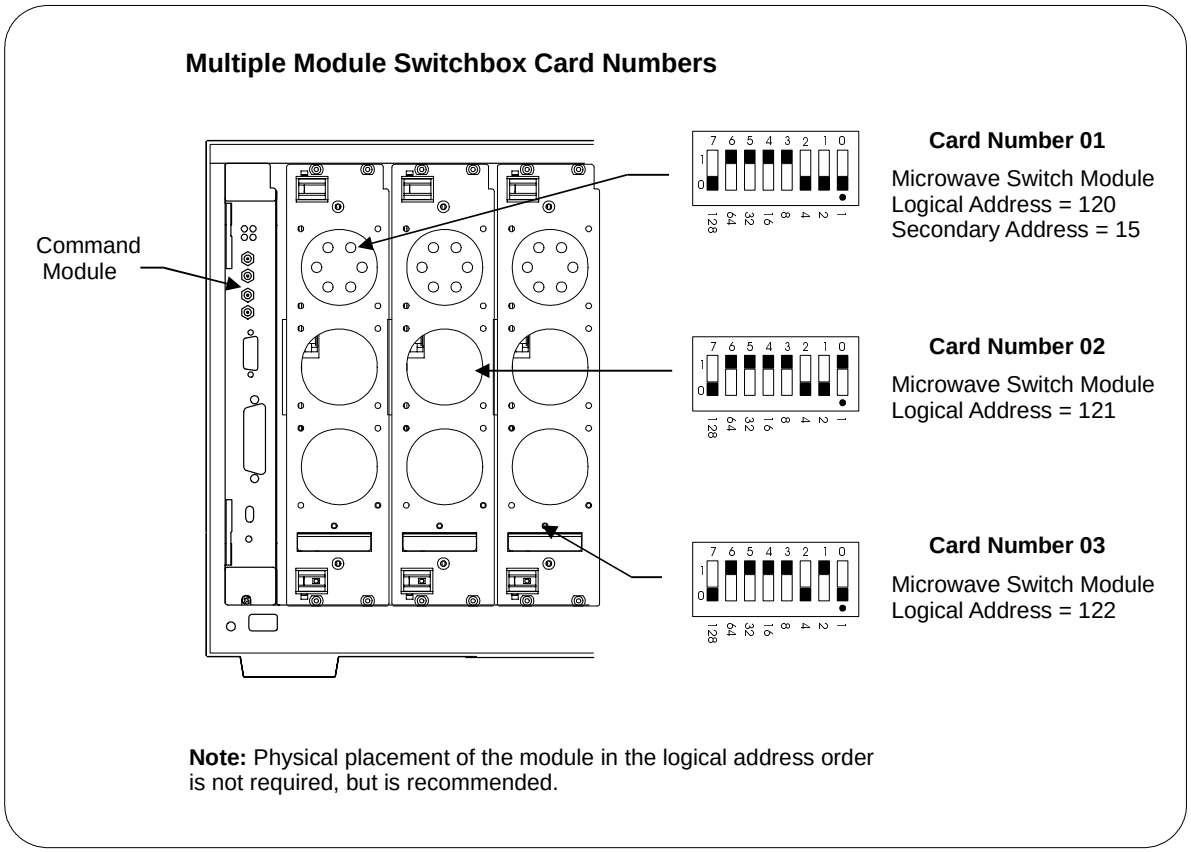


Figure 1-5. Card Numbers in a Multiple-modules Switchbox

Channel Number

The channel number (*gs* of the *channel_list*) identifies which RF port within a microwave switch will be addressed. The channel number depends on the type of the microwave switch and the group number of the connector to which the switch is connected. Table 1-2 lists the valid channel numbers for the microwave switches. Note that the channel number does not directly correspond to the same port number of the switches. The maps of the port numbers of the switches to the channel numbers of the module are shown in Table 1-3. For more information about these microwave switches that can be used with the E8483A module, refer to the corresponding Technical Data Sheet.

Table 1-2. Channel Number for Microwave Switches

Switch Type	Channel Number
87104A,B,C	g (group number) = 0-5; s (switch RF port number)= 0-3
87106A,B,C	g (group number) = 0-5; s (switch RF port number)= 0-5

Table 1-3. Map of Channel Numbers to Port Numbers

Switch Type	s = 0	s = 1	s = 2	s = 3	s = 4	s = 5
87104A,B,C [a]	port 2	port 3	port 5	port 6	-	-
87106A,B,C	port 1	port 2	port 3	port 4	port 5	port 6

a. Ports 1 and 4 are not available for the Agilent 87104A/B/C, so the range of channel number "s" is from 0 to 3.

For example, to address an 87104C SP4T Microwave Switch being connected to the 16-pin Group 2 connector of the E8483A driver board, the valid channel numbers can be 20 through 23. To connect port 3 to port C on the switch, execute the command: CLOSe (@121).

Programming with Step Attenuators

When the E8483A module is used to control the step attenuator, the attenuator is identified by the group number of the connector to which the attenuator is connected. Table 1-4 lists the most commonly used commands to control attenuators. The <*card_num*> can be 1 through 99 and the <*group_num*> can be 0 through 5. They are identical to those for addressing microwave switches as shown on Page 17. The attenuation range <*db*> depends on the type of the attenuators as listed in Table 1-5. For more information about the step attenuators that can be used with the E8483A module, refer to the corresponding Technical Data Sheet.

Table 1-4. Commonly Used SCPI Commands for Controlling Attenuators

SCPI Commands [a]	Commands Description
INPut:ATTenuation[:LEVel] <*card_num*>,<*group_num*>,<*db*>	Set attenuation value for the selected attenuator.
INPut:ATTenuation[:LEVel]? <*card_num*>,<*group_num*>	Query attenuation value set for the selected attenuator.

a. The meaning of the <card_num> and the <group_num> are identical to those for addressing microwave switches. <card_num> can be 1-99 and <group_num> can be 0-5. The range of <db> value depends on the type of the attenuator as shown in the table below.

Table 1-5. Attenuation Range of Various Attenuators

Attenuator Type	Range of <db> Value
Agilent 84904K, L	0 - 11 dB, 1 dB steps
Agilent 84906K, L	0 - 90 dB, 10 dB steps
Agilent 84907K, L	0 - 70 dB, 10 dB steps

For example, to set 20 dB attenuation for an 84906K Step Attenuator being connected to the Group 2 connector (10-pin) of an E8483A module, execute the command: INP:ATT 1, 2, 20.

Initial Operation

Use the following example programs to verify the initial operation on the E8483A Microwave Switch/Attenuator Driver module. To run the programs, an Agilent E1406A command module is required. Also, you should download the Agilent E8483A SCPI driver into the E1406A command module and have the Agilent SICL Library, the Agilent VISA extensions, and an Agilent 82350 GPIB module installed and properly configured in your PC.

In the examples, the computer interfaces to the mainframe via GPIB. The GPIB interface select code is 7, the GPIB primary address is 09, and the E8483A 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: Module Identification (HTBasic)

This example program was written in HTBasic programming language. It verifies communication between the E1406A command module and the E8483A module. It resets and identifies the E8483A module.

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

100  END
```

Example: Module Identification (C/C++)

This example program was developed and tested in Microsoft® Visual C++ but should compile under any standard ANSI C compiler. It verifies communication between the E1406A command module and the E8483A module. It resets and identifies the E8483A module.

```
#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 E8483A;                    /* 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,&E8483A);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpen() returned 0x%x\n",errStatus);
      return errStatus;}
```

```
    /* Reset the module */
    errStatus = viPrintf(E8483A, "*RST;*CLS\n");
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
        return errStatus;}

    /* Query the module ID string */
    errStatus = viQueryf(E8483A, "*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(E8483A, "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(E8483A, "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 (E8483A);
    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 Module

About This Chapter

This chapter shows how to configure the E8483A module for use with microwave switches and attenuators, install it in a mainframe, and connect field wiring to the module. Chapter contents include:

Warnings and Cautions

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

Caution **MAXIMUM POWER. The maximum power that may be applied to any SMA input connector is 1 W (CW).**

CONNECTING +24V. For the Microwave Switch, the mainframe backplane +24V is fused at 1.2 A. The total current drawn by all coaxial switches connected to the Microwave Switch module must not exceed the fuse rating of the supplies (mainframe and/or external) used.

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

Setting the Logical Address

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

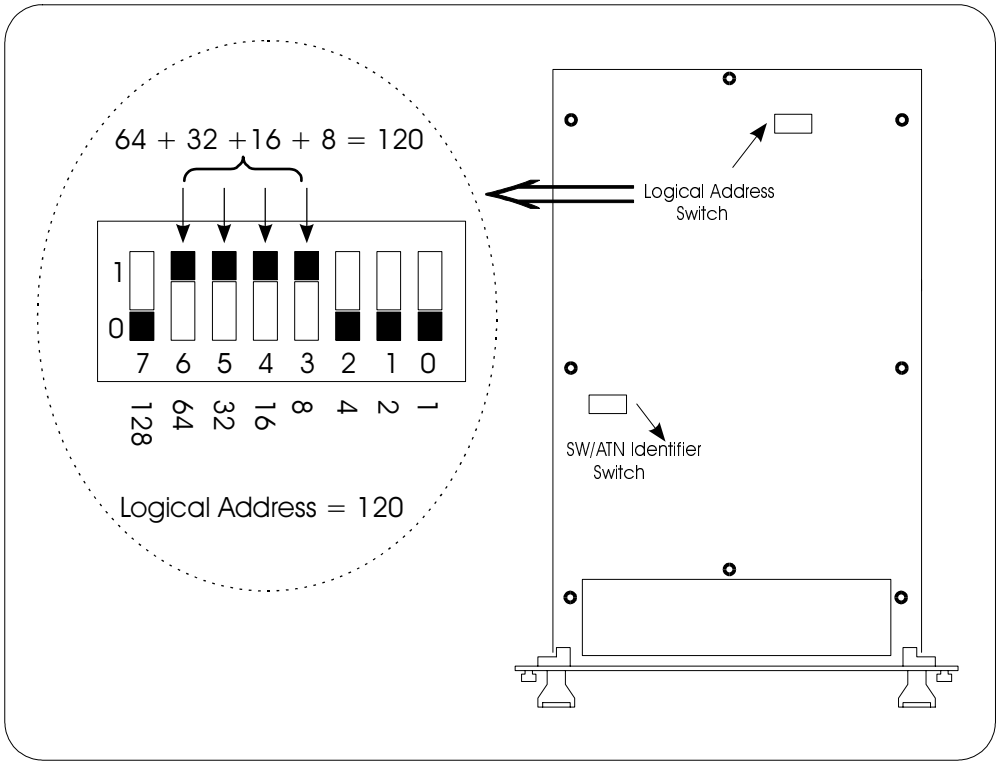


Figure 2-1. Setting the Logical Address Switch

Setting the Interrupt Priority

The E8483A 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, the default interrupt priority level should 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 59 of this manual for more details of the related SCPI commands. For more information about register writing, see “Register-Based Programming” on page 95 of this manual.

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.*

Setting the SW/ATN Identifier Switch

There is an SW/ATN Identifier switch (8-bits) on the module used to identify whether a microwave switch or a step attenuator is to be controlled by each group of the drive circuits of the module. When shipped from the factory, the E8483A module is set to be controlling six microwave switches. If the attenuators are to be controlled by the E8483A, you must change the corresponding bits of the SW/ATN Identifier switch to what each connector is really connected with. Figure 2-2 shows the SW/ATN Identifier switch position and setting information.

NOTE *DO NOT connect a microwave switch and an attenuator to the connectors within the same group. That is, if a microwave switch is connected to the 16-pin connector (i.e. Group 0), then you can not connect an attenuator to the 10-pin connector of the same group (i.e. Group 0). Otherwise, there may cause damage to the module. See "Guidelines for Connecting Switches/Attenuators" on page 28 of this manual for details.*

NOTE *Make sure the settings of the SW/ATN Identifier switch are consistent with what the connectors are really connected with. If a bit of the SW/ATN switch is set for a microwave switch but the corresponding connector is connected with an attenuator (or vice versa), the interface will not be able to identify it correctly, and an error will occur.*

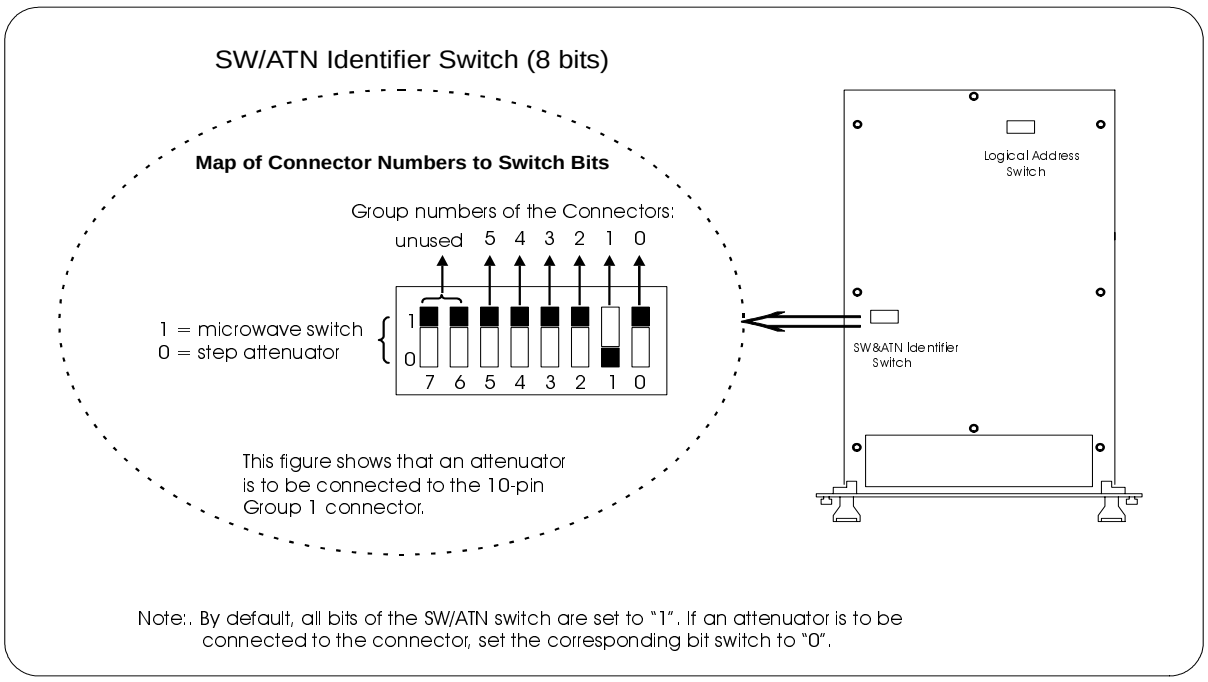
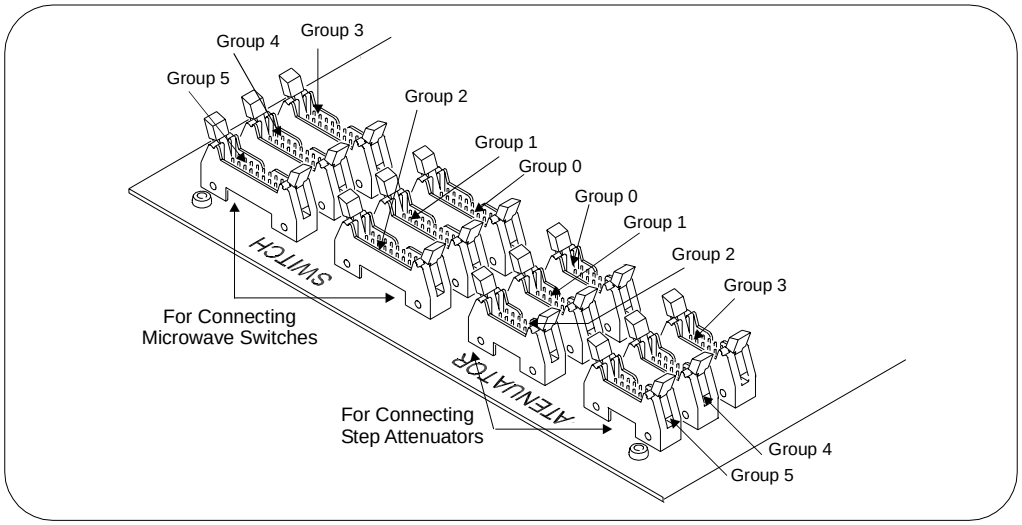


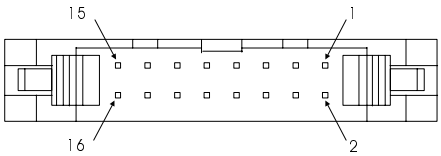
Figure 2-2. Setting Switch/Attenuator Identifier

Connectors Pinout

The six 16-pin ribbon cable connectors (Groups 0-5) on the PC board of the E8483A module accommodate the need for connecting microwave switches, and six 10-pin ribbon cable connectors (Groups 0-5) for connecting step attenuators. See Figure 2-3 for these connectors location and pinout information. In addition, the ribbon cables are also provided along with the module, which makes the connections more easily.



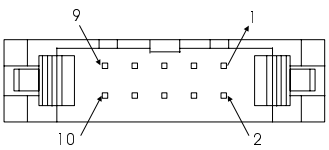
One of the Six Connectors for Microwave Switches



Pins	Function
1	+24 Vdc
2	Ind. Comm.
3	Path 1*
4	Ind. 1
5	Path 2
6	Ind. 2
7	Path 3
8	Ind.3
9	Path 4*
10	Ind. 4
11	Path 5
12	Ind. 5
13	Path 6
14	Ind. 6
15	Common Ground
16	Open all paths

* Paths 1 and 4 not connected for 87104A,B,C

One of the Six Connectors for Attenuators



Pins	Function
1	Section 1 Thru Line
2	Section 1 Attn Card
3	Common Ground
4	Section 3 Thru Line
5	Section 2 Thru Line
6	Section 4 Thru Line*
7	Section 4 Attn Card*
8	Section 2 Attn Card
9	Section 3 Attn Card
10	+24 Vdc

* Section 4 not connected for 84907K/L.

Figure 2-3. Group 0 - 5 Connectors Pinout

Guidelines for Connecting Switches/Attenuators

The E8483A module is not supplied with any microwave switches or attenuators which should be ordered separately. The recommended microwave switches and attenuators are listed in Table 2-1 and Table 2-2. Use the following guidelines before making connection:

- Refer to Table 2-1 and Table 2-2 respectively, to order the microwave switches and attenuators that are applicable for your application. For more information about the listed products, refer to their corresponding Technical Data Sheet.
- Determine the number of switches or attenuators to be connected. Up to six in any combination can be controlled by the module.
- Do not connect a microwave switch and a step attenuator to the connectors within the same group. That is, if you have connected a switch to the 16-pin Group 0 connector, then you can not connect an attenuator to the 10-pin Group 0 connector.
- Microwave switches can be either installed on, or externally connected to the module via 16-pin ribbon cables. One end with the female connector is for connecting to the driver board, and the other end with male connector is for connecting to the switch. See Figure 2-3 for more information on the connectors pinout. Note that only three switches can be installed on the module.
- Attenuators can be externally connected to the module via 10-pin ribbon cables. One end with the female connector is for connecting to the driver board, and the other end with male connector is for connecting to the attenuator. See Figure 2-3 for more information on the connectors pinout.

Table 2-1. Recommended Microwave Switches

Part Number	Frequency	Ports
87104A	DC - 4 GHz	single-pole, four-through (SP4T)
87104B	DC - 20 GHz	single-pole, four-through (SP4T)
87104C	DC - 26.5 GHz	single-pole, four-through (SP4T)
87106A	DC - 4 GHz	single-pole, six-through (SP6T)
87106B	DC - 20 GHz	single-pole, six-through (SP6T)
87106C	DC - 26.5 GHz	single-pole, six-through (SP6T)

Table 2-2. Recommended Step Attenuators

Part NO.	Frequency	Attenuation	Step
84904K	DC - 26.5 GHz	0 - 11 dB	1 dB
84904L	DC - 40 GHz	0 - 11 dB	1 dB
84906K	DC - 26.5 GHz	0 - 90 dB	10 dB
84906L	DC - 40 GHz	0 - 90 dB	10 dB
84907K	DC - 26.5 GHz	0 - 70 dB	10 dB
84907L	DC - 40 GHz	0 - 70 dB	10 dB

Installing/Connecting Microwave Switches/Attenuators

After selecting the microwave switches/attenuators and determining which connectors that they will be connected to, you may need to install the switches (up to three) on the module, or externally connect the attenuators or switches (if more than three switches are to be used in your application, or installing switches on the module is not desired) to the module. To install switches on the module, perform the procedures (step 1, step 2 and step 4) as shown in Figure 2-4. To connect attenuators or switches externally to the module, perform the procedures (step 1, step 3 and step 4) as shown in Figure 2-4.

NOTE *Make sure the SW/ATN Identifier switch has been set correctly after installation. See "Setting the SW/ATN Identifier Switch" on page 26 for more details.*

NOTE *Do not connect a microwave switch and an attenuator to the connectors labeled with the same group number.*

NOTE *Totally, six switches/attenuators can be connected to the E8483A Driver module.*

Step 1: Remove the Shield Metal

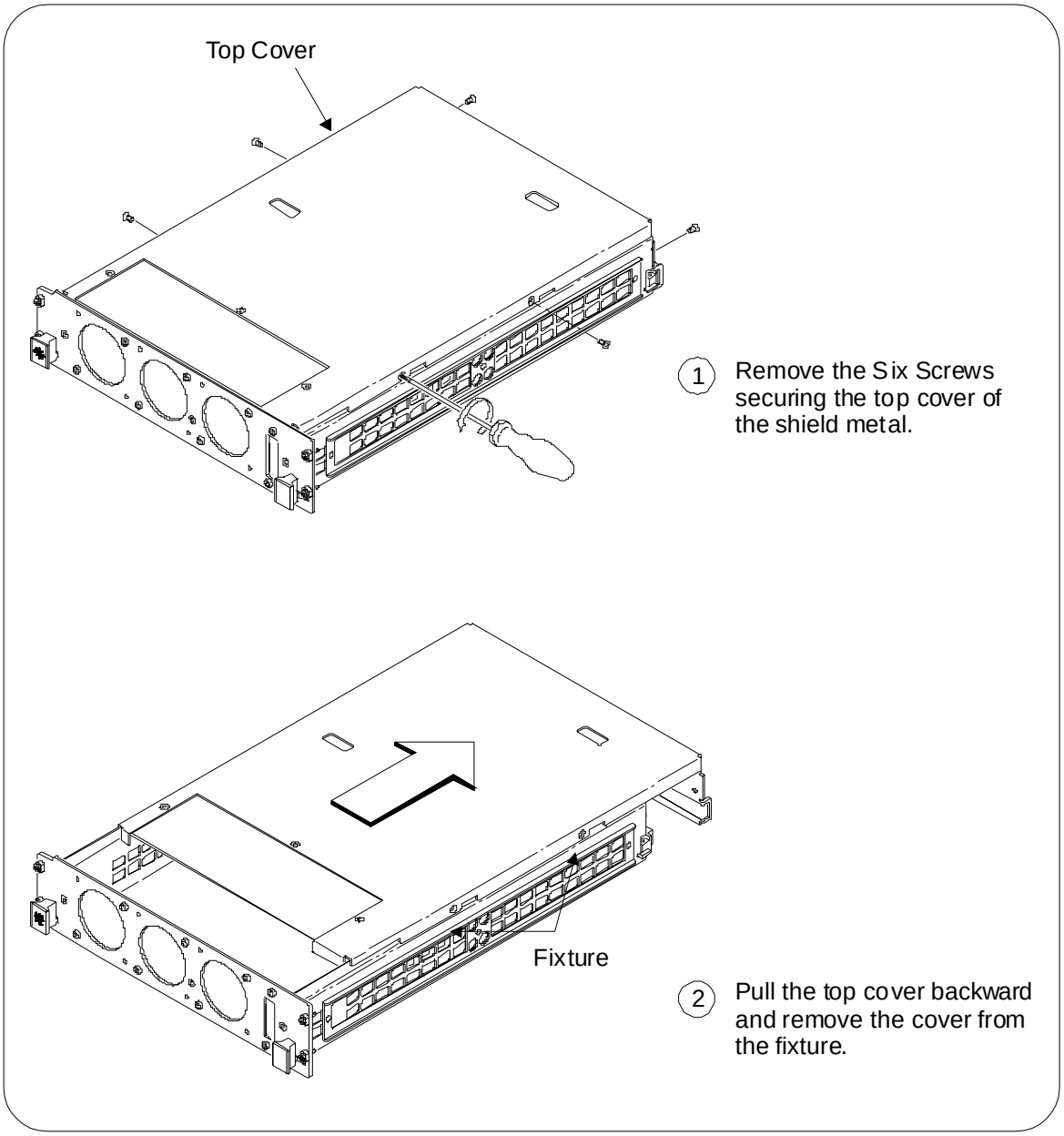


Figure 2-4. Install/Connect Microwave Switches or Attenuators
(*Continued on Next Page)*

Step 2: Install a Microwave Switch on the Module

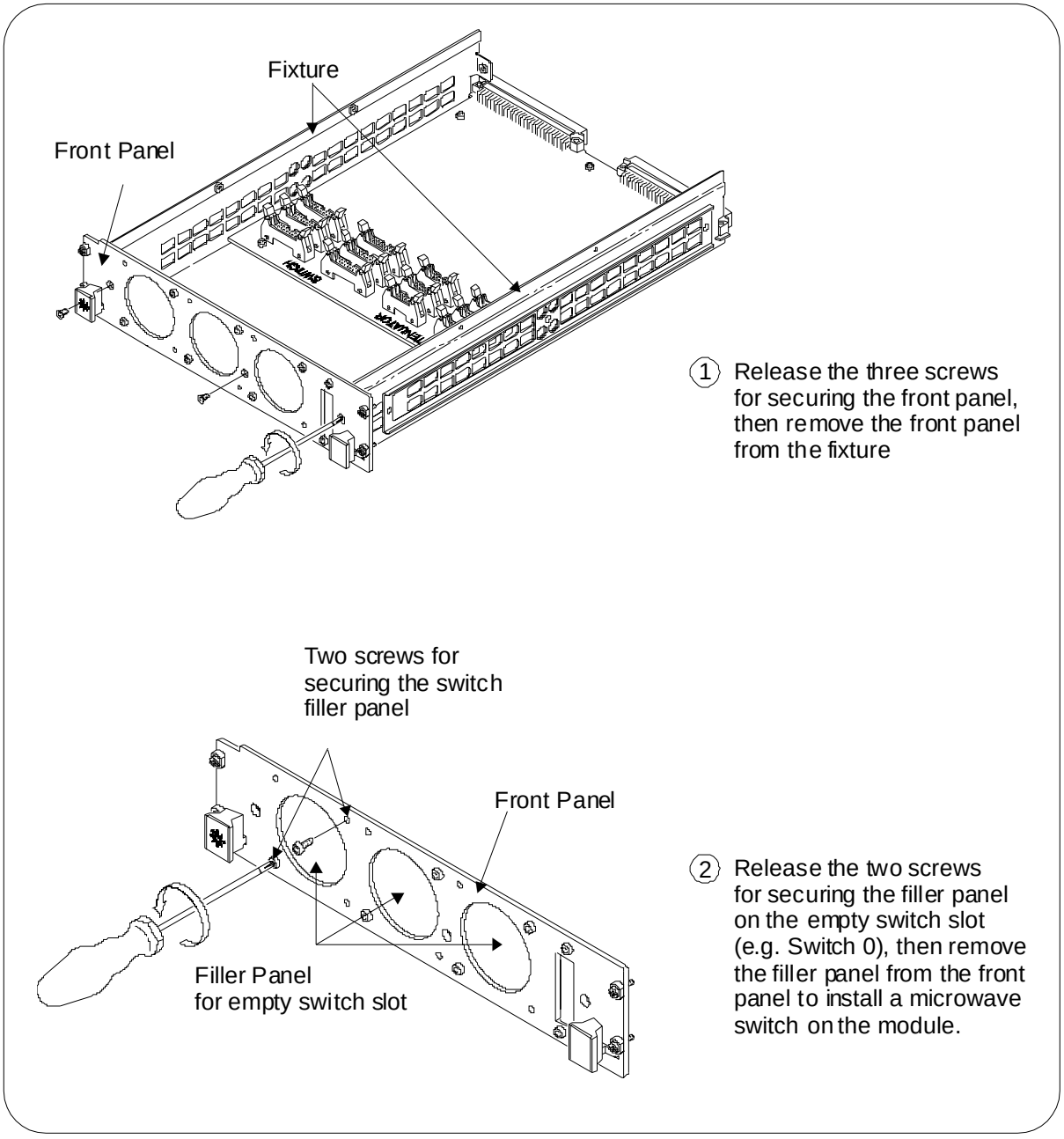
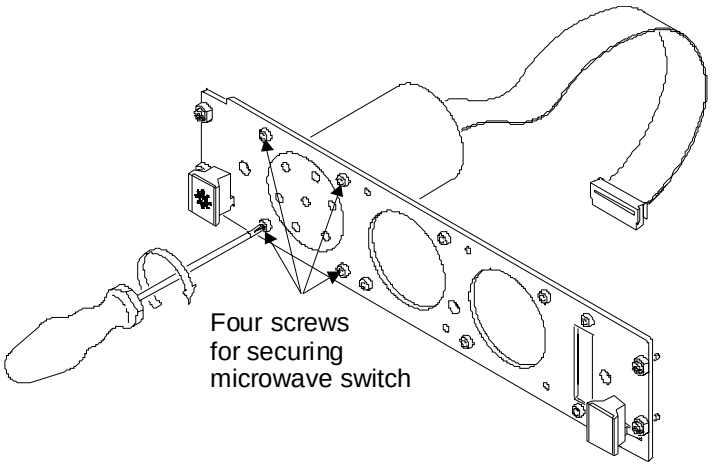
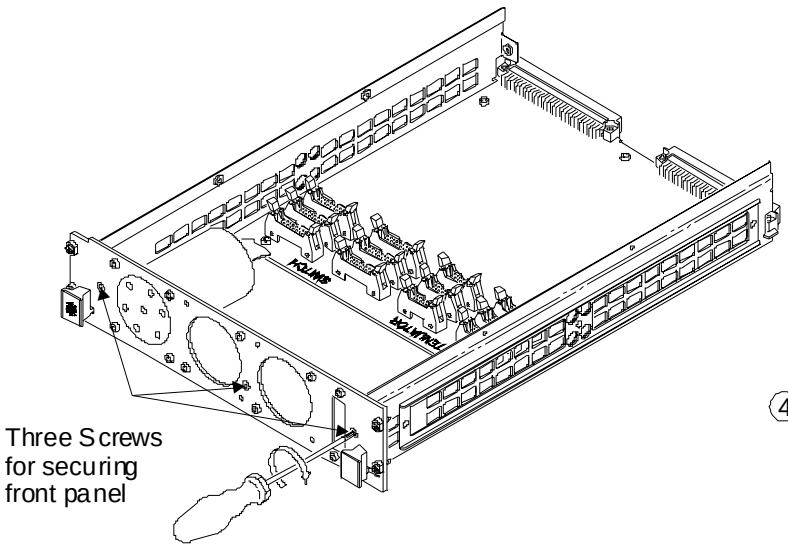


Figure 2-4. Install/Connect Microwave Switches or Attenuators
(*Continued on Next Page*)

Step 2: Install a Microwave Switch on the Module (*Continued*)



3 Position the microwave switch behind the panel cut-out, then secure the switch to the front panel of the module with the four screws.



4 Tighten the three screws (removed in Step 1) to reattach the front panel to the fixture.

Figure 2-4. Install/Connect Microwave Switches or Attenuators
(*Continued on Next Page*)

Step 2: Install a Microwave Switch on the Module (*Continued*)

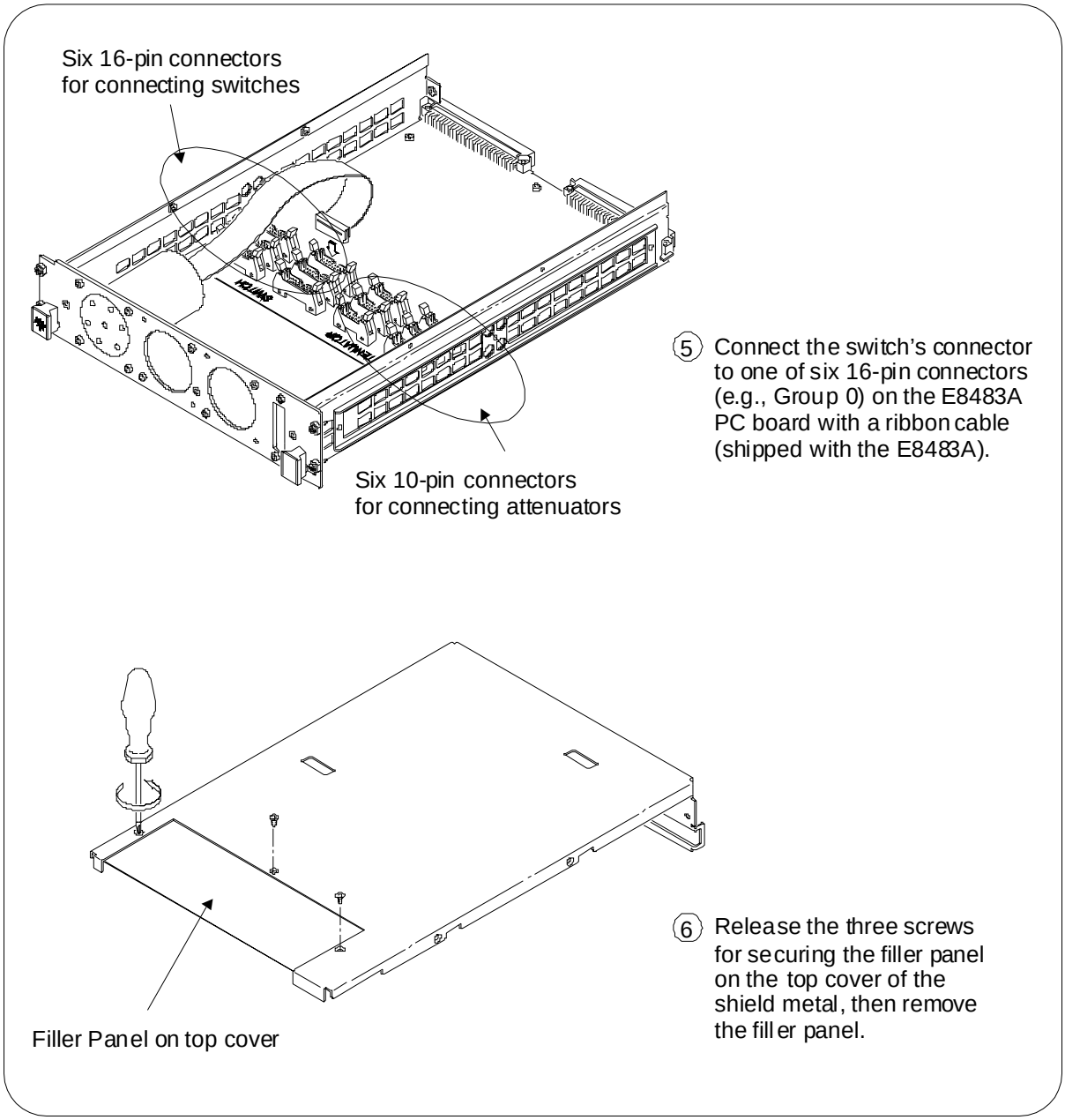


Figure 2-4. Install/Connect Microwave Switches or Attenuators
(*Continued on Next Page*)

Step 3: Externally Connect an Attenuator or a Switch (if required)

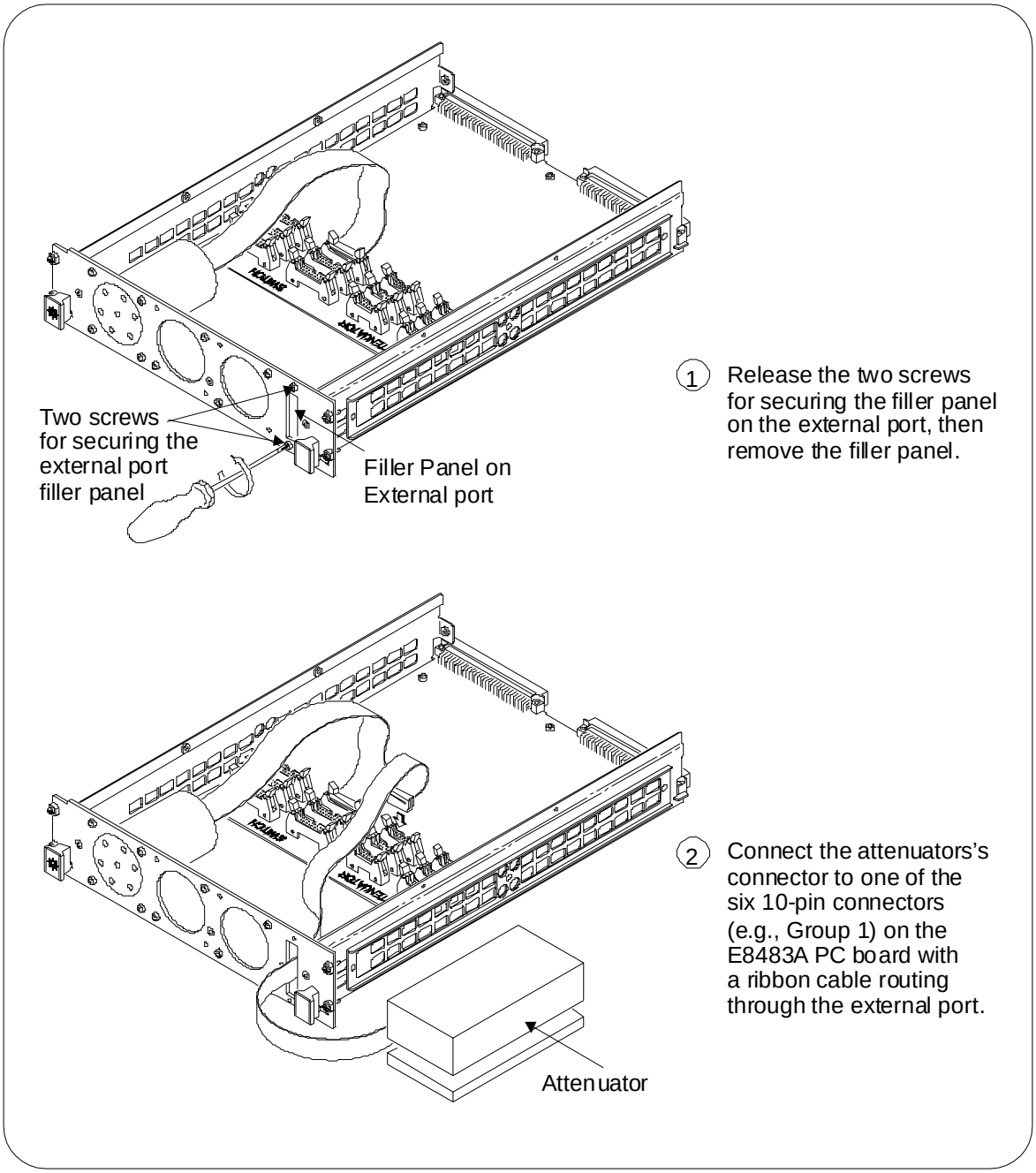


Figure 2-4. Install/Connect Microwave Switches or Attenuators
(*Continued on Next Page*)

Step 4: Replace the Shield Metal

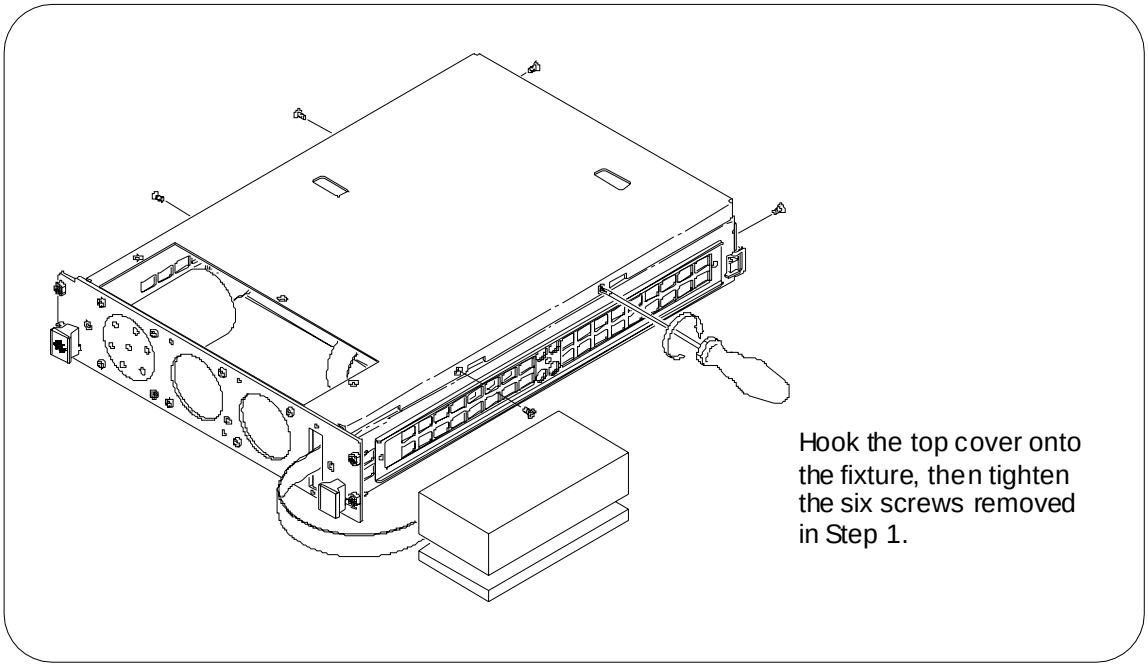


Figure 2-4. Install/Connect Microwave Switches or Attenuators

NOTE *If no microwave switch is installed on the module, it is not necessary to remove the filler panel on the top cover of the shield metal.*

NOTE *For external connection, the cables can also be routed through the empty switch slot if available.*

Notes:

Chapter 3
Using the E8483A

About This Chapter

This chapter uses typical examples to show how to use the Agilent E8483A Microwave Switch/Step Attenuator Driver module. See Chapter 4, "Command Reference" for the details of SCPI commands information. Chapter contents are:

All example programs in this chapter were developed on an external PC using HTBasic or Visual C/C++ as the programming language. They are tested with the following system configuration:

- An E1406A command module and an E8483A Microwave Switch/Attenuator Driver module are installed in the mainframe.
- The computer is connected to the E1406A command module via GPIB interface. The GPIB select code is 7, the GPIB primary address is 09, and the E8483A module is at logical address 120 (secondary address = 120/8 = 15).
- The E8483A SCPI driver had been downloaded into the E1406A command module.
- The SICL Library, the VISA extensions, and an Agilent 82350 GPIB card had been installed and properly configured in the computer.

Refer to the *Agilent E1406A Command Module User's Guide* for more addressing information. For more details on the related SCPI commands used in this chapter, see Chapter 4 of this manual.

NOTE *Do not do register writes if you are controlling the module by a high level driver such as SCPI or VXIplug&play. This is because the driver will not know the module state and an interrupt may occur causing the driver and/or command module to fail.*

Module Commands Summary

Table 3-1 explains some of the SCPI commands used with the microwave switches. Table 3-2 lists some of the SCPI commands used with the attenuators. Refer to Chapter 4 for more information on these commands.

Table 3-1. SCPI Commands Used for Microwave Switches

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*>	Open the channels in the channel list.
[ROUTe:]OPEN? <*channel_list*>	Query the state of the channels in the channel list.
[ROUTe:]SCAN <*channel_list*>	Define the channel list to be scanned. Channels specified are closed one at a time.
INITiate[:IMMediate]	Start the scan sequence and close the first channel in the channel list.
OUTPut[:STATe] ON	Enable "Trig Out" port on the mainframe to output pulses.
TRIGger:SOURce <*source*>	Select the trigger source to advance the scan.
SYSTem:COPTion <*card_num*>, <type> [a]	Set the type of microwave switches/attenuators being connected.

a. Each time the <type> is changed, you should recycle the E1406A command module to make the change effective.

Table 3-2. SCPI Commands Used for Attenuators

Commands	Description
INPut:ATTenuation[:LEVel] <card_num>,<group_num>,<db_value>	Set the attenuation value for the selected attenuator.
INPut:ATTenuation[:LEVel]? <card_num>,<group_num>	Query the attenuation value set for the selected attenuator.
SYSTem:COPTion <*card_num*>, <type> [a]	Set the type of microwave switches/attenuators being connected.

a. Each time the <type> is changed, you should recycle the E1406A command module to make the change effective.

Power-On and Reset Conditions

When the E8483A module is powered up or *RST (reset), all relays on the module are open and the current channel list for scanning is invalidated. Table 3-3 lists the parameters and default values for the functions.

Table 3-3. *RST Default Conditions

Parameter	Default	Description
ARM:COUNt	1	Number of scanning cycles is 1.
TRIGger:SOURce	IMM	Advances through a scanning list automatically.
INITiate:CONTinuous	OFF	Continuous scanning is disabled.
OUTPut:ECLTrg*n*[:STATe]	OFF	Trigger output from ECL trigger line is disabled.
OUTPut[:EXTernal][:STATe]	OFF	Trigger output from "Trig Out" port is disabled.
OUTPut:TTLTrg*n*[:STATe]	OFF	Trigger output from TTL trigger line is disabled.
INPut:ATTenuation[:LEVel]	0 dB	The attenuation value is 0 dB.

Module Identification

The following programs use the *RST, *CLS, *IDN?, SYST:CTYP?, and SYST:CDES? commands to reset and identify the E8483A module.

Example: Identifying Module (HTBasic)

```
10  DIM A$[50], B$[50], C$[50]          ! Dimension three string variables.
20  OUTPUT 70915; "*RST; *CLS"          ! Reset the module and clear status registers.
30  OUTPUT 70915; "*IDN?"               ! Query for module identification.
40  ENTER 70915; A$                     ! 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 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 E8483A;                   /* 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,&E8483A);
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viOpen() returned 0x%x\n",errStatus);
        return errStatus;}

     /* Reset the module and clear the status registers */
    errStatus = viPrintf(E8483A, "*RST;*CLS\n");
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
        return errStatus;}

     /* Query the module ID string */
    errStatus = viQueryf(E8483A, "*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(E8483A, "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(E8483A, "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 (E8483A);
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;
}
```

Setting Microwave Switches/Attenuators Type

Before operating the switches and/or attenuators being connected to the E8483A Driver module, you should set the type of the switches and/or attenuators according to the actual hardware configuration. Use the SYST:COPT *<card_num>, <type>* command to set their types, and use SYST:COPT? *<card_num>* command to verify the settings. Refer to Chapter 2 of this manual for the hardware configuration. For more details of the related SCPI commands, refer to Chapter 4 of this manual.

The following programs are written in HTBasic and C/C++ programming languages respectively. Assuming an Agilent 87106C Microwave Switch is being connected to the 16-pin Group 0 connector, an Agilent 84906K Step Attenuator is connected to the 10-pin Group 3 connector, and all other connectors are empty.

Example: Setting Switches/ Attenuators Type (HTBasic)

```
10  DIM OPT$[100]                          ! Dimension a string variables.
20  OUTPUT 70915; "*RST; *CLS"             ! Reset the module and clear
                                             Status Register.
30  OUTPUT 70915; "SYST:COPT 1, ""AGT87106, UNKNOWN,
      UNKNOWN,AGT84906, UNKNOWN, UNKNOWN"""
                                           ! Set the microwave
                                             switches/attenuators
                                             connected.
40  OUTPUT 70915; "SYST:COPT? 1"           ! Query for the microwave
                                             switches/attenuators type.
50  ENTER 70915; OPT$                      ! Enter the result into OPT$.
60  PRINT OPT$                             ! Print the contents of the
                                             variable OPT$.
70  END
```

Example: Setting Switches/ Attenuators Type (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 E8483A;                    /* Module session  */
   char optType[256];                   /* Switches/attenuators 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,&E8483A);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpen() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Reset the module and clear status registers */
   errStatus = viPrintf(E8483A, "*RST;*CLS\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Set the type of the microwave switches/attenuators being connected */
   errStatus = viPrintf(E8483A, "SYST:COPT 1, \"AGT87106,UNKNOWN,
      UNKNOWN,AGT84906,UNKNOWN,UNKNOWN\"\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Query the switches/attenuators type */
   errStatus = viQueryf(E8483A, "SYST:COPT? 1\n", "%t", optType);
   if (VI_SUCCESS > errStatus) {
      printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
      return errStatus;}
   printf("Connected Switches/Attenuators are %s\n", optType);

    /* Close the module instrument session */
   errStatus = viClose (E8483A);
   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 Microwave Switch Channels

Both microwave switches and attenuators can be controlled by the E8483A Microwave Switch/Attenuator Driver module. For microwave switches operation, you can connect or disconnect a signal by closing or opening a specific channel relay on the module. One port per switch can be connected to its C Port at a time, so only one channel relay in each group of the module can be closed at a time.

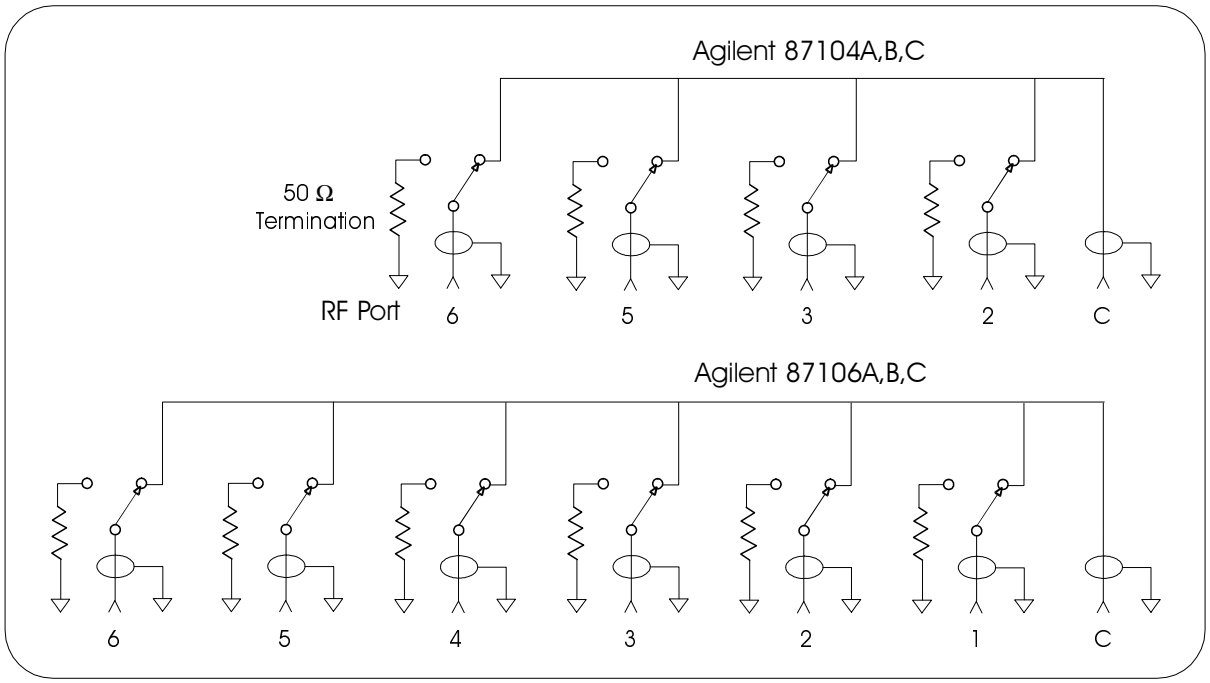


Figure 3-1. Port Numbers of the Microwave Switches

Use CLOSe <*channel_list*> and OPEN <*channel_list*> to close/open channel relays. The *channel_list* has the form (@ccgs) for a single channel, and (@ccgs,ccgs,...) for two or more channels. Where,

cc = card number (01-99)
g = group number (0-5)
s = 0-3 for Agilent 87104A/B/C microwave switch
    0-5 for Agilent 87106A/B/C microwave switch

Because of the ribbon cable configuration, the channel number of the module does not directly correspond to the same port number of the microwave switch. Table 3-4 maps the port numbers to the corresponding channel numbers.

Table 3-4. Map of Channel Numbers to Port Numbers

Switch Type	s = 0	s = 1	s = 2	s = 3	s = 4	s = 5
87104A,B,C [a]	port 2	port 3	port 5	port 6	-	-
87106A,B,C	port 1	port 2	port 3	port 4	port 5	port 6

a. Ports 1 and 4 are not available for the Agilent 87104A/B/C, so the range of channel number *"s"* is from 0 to 3.

For example, to connect C to port 3 on the Agilent 87104C Single-pole Four-throw Microwave Switch which is being connected to the 16-pin Group 1 connector of the E8483A module, use CLOSe (@111).

The following example programs are written in HTBasic and Visual C/C++ programming languages respectively. Assuming an Agilent 87106C Microwave Switch is being connected to the 16-pin Group 0 connector and has been configured correctly. This example illustrates how to connect or disconnect a signal by closing or opening a specific channel. It connects C port to port 3 of the switch by closing channel 102 of the module, then query to see whether the channel is closed. The result is returned to the computer and displayed (1 = channel closed, 0 = channel open).

Refer to Chapter 2 of this manual for more information on the hardware configuration. For more details of the SCPI commands used in the following example programs, refer to Chapter 4 of this manual. For more information about the microwave switches used, refer to the corresponding Technical Data Sheet.

Example: Closing a Switch Channel (HTBasic)

```
10  DIM A$[20]                              ! Dimension a string variable.
20  OUTPUT 70915; "*RST; *CLS"              ! Reset the module and clear
                                              Status Registers.
30  OUTPUT 70915; "ROUT:CLOS (@102)" ! Close channels 102 to connect
                                              C port to the port 3 on the
                                              switch.
40  OUTPUT 70915; "ROUT:CLOS? (@102)"! Query  to see whether channel
                                              102 is closed.
50  ENTER 70915; A$                         ! Enter the result into A$.
60  PRINT A$                                ! "1" returned indicates
                                              it is closed.
70  END
```

Example: Closing a Switch Channel (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 E8483A;                      /* 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,&E8483A);
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viOpen() returned 0x%x\n",errStatus);
        return errStatus;}

      /* Reset the module and clear status registers */
    errStatus = viPrintf(E8483A, "*RST;*CLS\n");
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
        return errStatus;}

      /* Close channel 02 of  the module*/
    errStatus = viPrintf(E8483A, "CLOS (@102)\n");
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
        return errStatus;}

      /* Query closure state of channel 02 */
    errStatus = viQueryf(E8483A, "ROUT:CLOS? (@102)\n", "%t", ch_stat);
    if (VI_SUCCESS > errStatus) {
        printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
        return errStatus;}
    printf("Channel 102 closed state is: %s\n", ch_stat);

      /* Close the module instrument session */
    errStatus = viClose (E8483A);
    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;
}
```

Scanning Microwave Switch Channels

For the Microwave switches, scanning channels consists of closing a specified set of channels, one at a time. You can scan any combination of channels for a single-module or a multiple-module switchbox. Single, multiple, or continuous scanning modes are available. Use SCAN <*channel_list*> to specify a set of channels to be scanned. Use ARM:COUNt <*number*> to set multiple/continuous scans (from 1 to 32,767 scans). Use TRIGger:SOURce <source> to select a trigger source to advance the scan. Use INITiate[:IMMediate] to start the scanning cycle. For more information about these SCPI commands, see Chapter 4 of this manual.

The following example programs are written in HTBasic and Visual C/C++ programming languages respectively. Assuming an Agilent 87106C Microwave Switch is being connected to the 16-pin Group 0 connector of the module and has been configured correctly. This example sets IMMediate (internal) triggering. The scan is automatically advanced through the specified channels list (100 through 105). By monitoring bit 7 of the Status Byte Register to determine when the scanning cycle is complete.

Refer to Chapter 2 of this manual for more information on the hardware configuration. For more details of the SCPI commands used in the following example programs, refer to Chapter 4 of this manual.

Example: Scanning Switch Channels with Internal Trigger (HTBasic)

```
10  OUTPUT 70915; "*RST;*CLS"          ! Reset the module and clear the
                                         status registers.

20  OUTPUT 70915; "STATUS:OPER:ENABLE 256"
                                       ! Enable Scan Complete Bit.

30  OUTPUT 70915; "TRIG:SOUR IMM"      ! Set the module for internal
                                         triggering.

40  OUTPUT 70915; "SCAN (@100:105)"    ! Set up channel list (100-105) to
                                         be scanned.

50  OUTPUT 70915; "INIT"               ! Start scan cycle and close the
                                         first scanned channel 100.
60  I =0
70  WHILE (I =0)                       ! Stay in loop until value
                                         returned from the command
                                         SPOLL (70915).
80      I = SPOLL (70915)
90      PRINT "Waiting for scan to complete..."
100 END WHILE

110 I = SPOLL (70915)                  ! "128" returned indicates scan
                                         has completed.
120 PRINT "Scan has completed: spoll = ";I
130 END
```

Example: Scanning Switch Channels with Internal Trigger (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 E8483A;                    /* Module session  */
   int scan;                            /* Variable for scan complete */

    /* 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,&E8483A);
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viOpen() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Set timeout value for the module */
   viSetAttribute (E8483A,VI_ATTR_TMO_VALUE,1000000);

    /* Reset the module and clear its status registers */
   errStatus = viPrintf(E8483A, "*RST;*CLS\n");
   if (VI_SUCCESS > errStatus) {
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Enable Scan Complete Bit */
   errStatus = viPrintf(E8483A, "STAT:OPER:ENAB 256\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Set trigger source to IMMediate for internal triggering */
   errStatus = viPrintf(E8483A, "TRIG:SOUR IMM\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}

    /* Specify a channel list for scanning */
   errStatus = viPrintf(E8483A, "SCAN (@100:105)\n");
   if(VI_SUCCESS > errStatus){
      printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
      return errStatus;}
```

```
    /* Start scan and close channel 100 */
    errStatus = viPrintf(E8483A, "INIT\n");
    if(VI_SUCCESS > errStatus){
        printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
        return errStatus;}

    /* Stay in loop until scan complete */
    for (; ;){
        errStatus = viQueryf(E8483A, "*STB?\n", "%d", &scan);
        printf("Waiting for scan to complete...\n");
        if (scan&0x80)
            break;}
    printf("Scan has completed!\n");

    /* Close the module instrument session */
    errStatus = viClose (E8483A);
    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;
}
```

Setting Attenuation Values for Attenuators

In addition to the Microwave Switches that can be used with the E8483A Driver module, the module can also be used to control step attenuators.

Use INPut:ATTenuation[:LEVel] <*card_num*>,<*group_num*>,<*db*> command to set the attenuation value for the specified attenuator. Use INPut:ATTenuation[:LEVel]? <*card_num*>,<*group_num*> command to query the settings. The range of the <db> value is dependent upon the type of the step attenuator as shown in Table 3-5.

Table 3-5. Attenuation Range of Various Attenuators

Attenuator Type	Range of <db> Value
Agilent 84904K, L	0 - 11 dB, 1 dB steps
Agilent 84906K, L	0 - 90 dB, 10 dB steps
Agilent 84907K, L	0 - 70 dB, 10 dB steps

For example, to set 20 dB attenuation for an 84906K Step Attenuator that is being connected to the 10-pin Group 2 connector of the E8483A module, execute the command: INP:ATT 1, 2, 20.

The following example programs assume that an Agilent 84906K Step Attenuator is being connected to the 10-pin Group 3 connector of the module and has been configured correctly. They are written in HTBasic and Visual C/C++ programming languages respectively. This example sets 40 dB attenuation value for the attenuator, then query to see whether the setting is correct. The result is returned to the computer and displayed.

Refer to Chapter 2 of this manual for more information on the hardware configuration. For more details of the SCPI commands used in the example programs, refer to Chapter 4 of this manual. For more information about the step attenuator used, refer to the corresponding Technical Data Sheet.

Example: Setting Attenuation Value (HTBasic)

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

30   OUTPUT 70915; "INP:ATT 1, 3, 40"        ! Set 40 dB attenuation for the
                                               attenuator being connected to
                                               Group 3 connector.
40   OUTPUT 70915; "INP:ATT? 1, 3"           ! Query the attenuation value.
50   ENTER 70915; A$                         ! Enter the result into A$.
60   PRINT "Attenuation Value is set to "; A$ ! Print the result.
70   END
```

Example: Setting Attenuation Value (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 E8483A;                        /* Module session */
   char att_val[10];                        /* Variable for attenuation
                                               values */

    /* 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,&E8483A);
if(VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

  /* Reset the module and clear the Status Registers*/
errStatus = viPrintf(E8483A, "*RST;*CLS\n");
if(VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

  /* Set 40 dB attenuation for the Group 3 Attenuator*/
errStatus = viPrintf(E8483A, "INP:ATT 1, 3, 40\n");
if(VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

  /* Query the attenuation value of the Group 3 Attenuator */
errStatus = viQueryf(E8483A, "INP:ATT? 1, 3\n", "%t", att_val);
if (VI_SUCCESS > errStatus) {
   printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
   return errStatus;}
printf("The attenuation value is: %s\n", att_val);

  /* Close the module instrument session */
errStatus = viClose (E8483A);
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;
}
```

Detecting Error Conditions

The SYSTem:ERRor? command queries the instrument's error queue for error conditions. If no error occurs, the 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 to query an attenuator with an illegal group number 6 (the valid group number is 0-5), then polls for the error message.

```
10   DIM Err_num$[256]                    ! Dimension a string variable.
20   OUTPUT 7091 5; "INP:ATT? 1,6"        ! Try to close an illegal channel.
30   OUTPUT 70915; ":SYST:ERR?"           ! Check for a system error.
40   ENTER 70915; Err_num$                ! Enter the error into Err_num$.
50   PRINT "Error: "; Err_num$            ! Print error -224, "Illegal
                                            parameter value".
60   END
```

Synchronizing the Instruments

This section shows how to synchronize the Microwave Switch/Attenuator Driver module with other instruments when making measurements. The following example assumes that an 87106C microwave switch is installed in the E8483A module (Group 0) and a signal to be measured by a multimeter is connected to the port 2 of the switch. The multimeter has the GPIB address of 70903 and the E8483A module has a logical address of 120 (GPIB address of 70915).

Example: Synchronizing the Instruments (HTBasic)

This example program was written in HTBasic language. It verifies that the switching is complete before the multimeter begins a measurement.

```
10   OUTPUT 70915; "*RST"                 ! Reset the module.
20   OUTPUT 70915; "CLOS (@101)"          ! Close a channel.
30   OUTPUT 70915; "CLOS? (@101)"         ! Verify that the channel is
                                            closed.
40   ENTER 70915; A
50   IF A=1 THEN
60      OUTPUT 70903; "MEAS:VOLT:DC?"     ! When channel is closed, make
                                            the measure.
70      ENTER 70903; Meas_value
80      PRINT Meas_value                  ! Print the measured value.
90   ELSE
100     PRINT "CHANNEL DID NOT CLOSE"
110  END IF
120  END
```

Recalling and Saving States

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

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

The *RCL <*numeric_state*> command recalls a previously saved state specified by the *numeric_state* parameter. If no *SAV was previously executed for the *numeric_state*, *RST default settings are used.

Querying the Module

All query commands end with a "?". The data is sent to the output buffer where you can retrieve it into your computer. The following summarizes some of the query commands you can use to obtain the specific information of the module. See Chapter 4 for more details of these commands.

Channel Closed:	CLOS?
Channel Open:	OPEN?
Attenuation Value:	INP:ATT?
Module Description:	SYST:CDES?
Module Type:	SYST:CTYP?
Switch/Attenuator Type:	SYST:COPT?
System error:	SYST:ERR?

Chapter 4
Command Reference

Using This Chapter

This chapter describes Standard Commands for Programmable Instruments (SCPI) and summarizes IEEE 488.2 Common (*) commands applicable to the E8483A module. See the *Agilent E1406A Command Module User's Manual* 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        *ESR <unmask>        *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>
    SCAN <channel_list>
```

[ROUTe:] is the root command, CLOSe and SCAN are the second level commands with <*channel_list*> as a parameter.

Command Separator

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

```
ROUTe:SCAN <channel_list>
```

Colons separate the root command from the second level (ROUTe:SCAN). 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 form or the entire command.

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

Implied Commands

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

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

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

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

Variable Commands

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

```
OUTPut:TTLTrgn
```

In this command, the "*n*" is replaced by a number (range from 0 to 7). No space is left between the command and the number because the number is part of the command syntax instead of a parameter.

Parameters

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

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.

Parameter Types	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.
String	Accepts an array of ASCII characters starting and ending with quotation marks ("). An example is the SYSTem:COPTion <*card_num*>, <*type*> command where <type> can be "AGT87106, AGT87104, AGT84906, UNKNOWN-ATN, UNKNOWN-SW, AGT84904".

Linking Commands

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

```
*RST;CLOS (@100)   or   TRIG:SOUR  HOLD;*TRG
```

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

```
ARM:COUN1;:TRIG:SOUR  EXT
```

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

```
ROUT:CLOS (@100);:ROUT:CLOS? (@100)
```

- or -

```
ROUT:CLOS (@100);CLOS? (@100)
```

SCPI Command Reference

This section describes the Standard Commands for Programmable Instruments (SCPI) commands for the Microwave Switch/Attenuator Driver module. Commands are listed alphabetically by subsystem and also within each subsystem.

NOTE *Some subsystem commands are used only for controlling Microwave Switches and some only for setting Step Attenuators. Unless specially specified, all commands apply for both devices.*

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

NOTE *This command is used only for Microwave Switches during scanning.*

Subsystem Syntax

ABORt

Comments

ABORt Actions: The ABORt command terminates the scan and invalidates the current channel list.

Stopping Scan Enabled Via Interface: When a scan is enabled via an interface, an interface clear command (CLEAR 7 or viClear () function in VISA) can be used to stop the scan. When the scan is enabled via the interface and TRIGger:SOURce BUS or HOLD is set, you can use ABORt command to stop the scan.

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

Example

Stopping a Scan with ABORt

This example stops a continuous scan in progress.

```
TRIG:SOUR  BUS          ! Set BUS as trigger source.
INIT:CONT  ON           ! Set continuous scanning.
SCAN  (@100:102)        ! Set channel list to be scanned.
INIT                    ! Start scan, close channel 00.
 .                       .
 .                       .
 .                       .
ABOR                    ! Abort scan in progress.
```

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

NOTE *This subsystem commands are used only for Microwave Switches.*

Subsystem Syntax

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

ARM:COUNt

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

Parameters

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

Comments

Number of Scans: Use only values between 1 to 32767, MIN, or MAX for the number of scanning cycles.

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

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

Example

Setting Ten Scanning Cycles

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

ARM:COUNt?

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

Parameters

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

Comments

Related Commands: INITiate[:IMMediate]

Example

Querying Number of Scanning Cycles

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

The **DIAGnostic** subsystem is used to control the module's interrupt capability, emergency protection capability, as well as the time interval between the two scanned channels. 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>
    :SCAN
        :DELay  <card_number>, <time_interval>
        :DELay?  <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 E8483A 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 1-7. Only one value can be set at one time. The default value is 1 (lowest interrupt level).

Related Commands: DIAGnostic:INTerrupt:[LINe]?

Example

Selecting Interrupt Line 1 for Module #1

```
DIAG:INT:LIN 1, 1          ! Set the interrupt line of Module #1 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 E8483A 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 Interrupt Line Used by Module #1

```
DIAG:INT:LIN 1, 1          ! Select interrupt line 1 for Module #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 E8483A 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 s - 0.064 s	0.02 second

Comments

The value of <*timer*> depends on the switches or attenuators being controlled by the E8483A module. For 87104/6A, B, C microwave switches, we highly recommend you set the time no less than 15 ms. For 84904/6/7K,L Step Attenuators, we highly recommend you set the time no less than 20 ms. If different types of switches and attenuators are to be controlled, set the <*timer*> to the maximum of them.

*RST does not change the selected time.

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

Example **Setting Module #1 Interrupt Timer to 20 ms**

```
DIAG:INT:TIM 1, 0.02          ! Set Module #1 Interrupt timer to 20 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 Interrupt Time Set for Module #1**

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

DIAGnostic:SCAN:DELay

DIAGnostic:SCAN:DELay <*card_number*>, <*delay_timer*> sets the amount of extra time the module will wait between opening one channel and closing the next in a scan operation. The *card_number* parameter specifies which E8483A in a multiple-module switchbox is to be set.

NOTE *This command is used only for Microwave Switches.*

Parameters

Name	Type	Range of Values	Default Value
<*card_number*>	numeric	1 - 99	N/A
<*delay_timer*>	numeric	0 - 6.5535 second	0 second

Example **Setting Module's Scan Delay Time to 5 ms**

```
DIAG:SCAN:DEL 1, 0.005        ! Module #1 will wait 5 ms between
                                opening one channel and closing the
                                next specified in the scan list.
```

DIAGnostic:SCAN:DELay?

DIAGnostic:SCAN:DELay? <*card_number*> queries the specified module and returns the delay time set by the DIAG:SCAN:DEL command.

NOTE *This command is used only for Microwave Switches.*

Example **Querying Scan Delay Time of Module #1**

```
DIAG:SCAN:DEL? 1              ! Query the scan delay time setting on 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 "0" if no error. Otherwise, return "-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 <card_num> | AUTO
        :CARD?
        [:STATe] <mode>
        [:STATe]?
```

DISPlay:MONitor:CARD

DISPlay:MONitor:CARD <***card_num***> **| 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
<*card_num*> | 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 presents an array of values indicating the open/close state of channels on 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 <card_num> | AUTO command to select the module.

Monitor Mode for an E8483A: When monitoring mode is turned ON, the closed channel number on each connected microwave switch or the attenuation value set for the connected attenuator will be displayed at the bottom of the terminal. "S" preceding the group number indicates a microwave switch is connected to the group and following the equation mark is the closed channel of the switch. "A" preceding the group number indicates an attenuator is connected to the group and following the equation mark is the attenuation value set for the attenuator. For example, the display:

Group: S0=5 S1=N S2=n A3=30 A4=10 A5=0

The example shows that three microwave switches are connected to the Group 0-2 connectors of the E8483A and Group 3-5 connectors are connected with attenuators. Channel 5 of Group 0 is closed to its common, all channels of Group 1 and Group 2 are open. The attenuation values for Groups 3 -5 are 30dB, 10dB and 0dB.

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

NOTE *This subsystem commands are used only for Microwave Switches.*

Subsystem Syntax

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

INITiate:CONTinuous

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

Parameters

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

Comments

Continuous Scanning Operation: Continuous scanning is enabled with the INITiate:CONTinuous ON or INITiate:CONTinuous 1 command. Sending the INITiate:IMMediate command closes the first channel in the channel list. Each trigger from the trigger source specified by the TRIGger:SOURce command advances the scan through the channel list. A trigger at the end of the channel list closes the first channel in the channel list and the scan cycle repeats.

Noncontinuous Scanning Operation: Noncontinuous scanning is enabled with the INITiate:CONTinuous OFF or INITiate:CONTinuous 0 command. Sending the INITiate:IMMediate command closes the first channel in the channel list. Each trigger from the trigger source specified by the TRIGger:SOURce command advances the scan through the channel list. A trigger at the end of the channel list opens the last channel in the list and the scanning cycle stops.

Stopping Continuous Scan: Refer to the ABORt command on page 56.

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

***RST Condition:** INITiate:CONTinuous OFF | 0

Example Enabling Continuous Scanning

This example enables continuous scanning of channels 00 through 03 of a single-module switchbox. Since TRIGger:SOURce IMMediate (default) is set, use an interface clear command (such as CLEAR 7) to stop the scan.

```
INIT:CONT  ON                 ! Enable continuous scanning.
SCAN  (@100:103)              ! Set channel list to be scanned.
INIT                          ! Start scanning, close channel 00.
```

INITiate:CONTinuous?

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

Example Querying Continuous Scanning State

```
INIT:CONT  ON                 ! Enable continuous scanning.
INIT:CONT?                    ! Query continuous scanning state. It
                                returns "1" (ON).
```

INITiate[:IMMediate]

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

Comments

Starting the Scanning Cycle: The INITiate[:IMMediate] command starts scanning by closing the first channel in the channel list. Each trigger received advances the scan to the next channel in the channel list. An invalid channel list definition generates an error (refer to [ROUTe:]SCAN command).

Stopping Scanning Cycles: Refer to the ABORt command on page 56.

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

Example Enabling a Single Scan

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

```
SCAN  (@100:103)              ! Set channels 00-03 to be scanned.
INIT                          ! Start scan, close channel 00 (use
                                immediate triggering).
```

The **INPut** subsystem sets the attenuation value for the selected attenuator.

NOTE *This subsystem commands are used only for Attenuators.*

Subsystem Syntax

```
INPut
    :ATTenuation[:LEVel] <card_num>, <group_num>, <db_value>,
    :ATTenuation[:LEVel]? <card_num>, <group_num>
```

INPut:ATTenuation[:LEVel]

INPut:ATTenuation[:LEVel] *<card_num>, <group_num>, <db_value>* is used to set an attenuation value for the selected attenuator.

NOTE *To make your attenuation setting effective, ensure that the attenuators type set by* SYST:COPT <*card_num*>, <*type*> command is completely consistent with your actual hardware configuration.

Parameters

Name	Type	Range of Values	Default Value
<*card_num*>	numeric	1 - 99	N/A
<*group_num*>	numeric	0 - 5	N/A
<*db_value*> [a]	numeric	0-11 dB, 1 dB steps (for 84904K,L) 0-90 dB, 10 dB steps (for 84906K,L) 0-70 dB, 10 dB steps (for 84907K,L)	0 dB

a. The range of <*db_value*> depends upon the type of the Attenuator.

Comments

Attenuation Value: Use only values within the range applicable for the selected attenuator. Refer to above table for details.

Related Commands: SYSTem:COPTion, SYSTem:COPTion?

***RST Condition:** The attenuation values of all attenuators are 0 dB.

Example

Setting 20 dB Attenuation for the 84906K Step Attenuator

```
SYST:COPT  1, "AGT84906,,,,,"      ! Set an Agilent 84906 connected to
                                     Group 0 connector.
(recycle the system)

INP:ATT 1, 0, 20                   ! Set 20 dB attenuation for the Attenuator
                                     connected to Card 1, Group 0.
```

INPut:ATTenuation[:LEVel]?

INPut:ATTenuation[:LEVel]? ***<card_num>, <group_num>*** returns the current attenuation value of the selected attenuator set by the INPut:ATTenuation[:LEVel] command.

Comments

Related Commands: INPut:ATTenuation[:LEVel]

Example

Querying the Attenuation Value Set for the Group 0 Attenuator

```
INP:ATT 1, 0, 20          ! Set 20 dB attenuation for the Attenuator
                            connected to Card 1, Group 0.

INP:ATT? 1, 0             ! Query the attenuation value set for the
                            Group 0 Attenuator, "20" will be
                            returned.
```

The **OUTPut** command subsystem selects the source of the output trigger generated when a channel is closed during a scan. The selected output can be enabled, disabled, or queried. The three available outputs are ECLTrg, TTLTrg trigger buses, and the "Trig Out" port on the command module's front panel (E1406A).

NOTE *This subsystem commands are used only for Microwave Switches.*

Subsystem Syntax

```
OUTPut
    :ECLTrgn       (:ECLTrg0 or :ECLTrg1)
        [:STATe]  <mode>
        [:STATe]?
    [:EXTernal]
        [:STATe]  <mode>
        [:STATe]?
    :TTLTrgn       (:TTLTrg0 through :TTLTrg7)
        [:STATe]  <mode>
        [:STATe]?
```

OUTPut:ECLTrg*n*[:STATe]

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

Parameters

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

Comments

Enabling ECL Trigger Bus: When enabled, a trigger pulse is output from the selected ECL Trigger bus line (0 or 1) each time a channel is closed during a scan. The output is a negative going pulse.

ECL Trigger Bus Line Shared by Switchboxes: Only one switchbox configuration can use the selected trigger at a time. When enabled, the selected ECL Trigger bus line (0 or 1) is pulsed by the switchbox each time a scanned channel is closed. To disable the output for a specific switchbox, send the OUTPut:ECLTrgn OFF or 0 command for that switchbox.

One Output Selected at a Time: Only one output (ECLTrg*n*, TTLTrg*n* or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if ECLTrg0 is the active output and ECLTrg1 is enabled, ECLTrg0 will become disabled and ECLTrg1 will become the active output.

Related Commands: [ROUTe:]SCAN, TRIGger:SOURce, OUTPut:ECLTrg*n*[:STATe]?

***RST Condition:** OUTPut:ECLTrg*n*[:STATe] OFF (disabled)

Example Enabling ECL Trigger Bus Line 0

```
OUTP:ECLT0:STAT 1          ! Enable ECL Trigger bus line 0 to
                             output pulse after each scanned
                             channel is closed.
```

OUTPut:ECLTrg*n*[:STATe]?

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

Example Querying ECL Trigger Bus Enable State

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

```
OUTP:ECLT1:STAT 1          ! Enable ECL Trigger bus line 1.
OUTP:ECLT1?                ! Query bus enable state.
```

OUTPut[:EXTernal][:STATe]

OUTPut[:EXTernal][:STATe] *<mode>* enables or disables the "Trig Out" port on the E1406A command module to output a trigger when a channel is closed during a scan.

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

Parameters

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

Comments

Enabling "Trig Out" Port: When enabled, a pulse is output from the "Trig Out" port each time a channel is closed during scanning. If disabled, a pulse is not output from the port after channel closures.

Output Pulse: The pulse is a +5 V negative-going pulse.

"Trig Out" Port Shared by Switchboxes: Only one switchbox configuration can use the selected trigger at a time. When enabled, the "Trig Out" port may is pulsed by the switchbox each time a scanned channel is closed. To disable the output for a specific switchbox, send the OUTP OFF or 0 command for that switchbox.

One Output Selected at a Time: Only one output (ECLTrg*n*, TTLTrg*n* or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if TTLTrg1 is the active output and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active output.

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

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

Example Enabling "Trig Out" Port

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

OUTPut[:EXTernal][:STATe]?

OUTPut[:EXTernal][:STATe]? queries the present state of the "Trig Out" port on the E1406A command module. The command returns "1" if the port is enabled or "0" if disabled.

Example Querying "Trig Out" Port State

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

OUTPut:TTLTrg*n*[:STATe]

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

Parameters

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

Comments

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

TTL Trigger Bus Line Shared by Switchboxes: Only one switchbox configuration can use the selected trigger at a time. When enabled, the selected TTL Trigger bus line (0 to 7) is pulsed by the switchbox each time a scanned channel is closed. To disable the output for a specific switchbox, send the OUTPut:TTLTrg*n* OFF or 0 command for that switchbox.

One Output Selected at a Time: Only one output (ECLTrg*n*, TTLTrg*n* or EXTernal) can be enabled at one time. Enabling a different output source will automatically disable the active output. For example, if TTLTrg1 is the active output and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active output.

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

***RST Condition:** OUTPut:TTLTrg*n*[:STATe] OFF (disabled)

Example Enabling TTL Trigger Bus Line 7

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

OUTPut:TTLTrg*n*[:STATe]?

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

Example Querying TTL Trigger Bus Enable State

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

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

The [ROUTe:] command subsystem are used for the E8483A module(s) to control switching and scanning operations on the Microwave Switch(es).

NOTE *This subsystem commands are used only for controlling Microwave Switches. See the appropriate Technical Data Sheet for the details of the Microwave Switches you selected.*

Subsystem Syntax

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

[ROUTe:]CLOSe

[ROUTe:]CLOSe *<channel_list>* closes Microwave Switch channels specified in the *channel_list*. The *channel_list* is in the form of (@ccgs), where cc = card number (01-99) and gs = channel number (microwave switch type dependant).

Parameters

Name	Type	Range of Values	Items
<*channel_list*>	numeric	1 - 99	card number (cc)
	numeric	g = 0 - 5, s = 0 - 3 (for 87104A,B,C) g = 0 - 5, s = 0 - 5 (for 87106A,B,C)	channel number (gs)

Comments

Closing Channels: To close:

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

Channel Range: The E8483A will accept and execute channel ranges (ccgs:ccgs) without generating an error, but the result is to close the last channel in each group within the range specified. For example, after CLOSe (@101:103,120:122) is executed, card #1 channels 03 and 22 would remain closed.

NOTE *Only one port (channel) in each group can be connected to its Common port at a time.*

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

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

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

Example Closing Microwave Switch Channels

This example closes channels 100 and 202 of a two-module switchbox (card numbers 01 and 02).

```
CLOS (@100,202)          ! Close channel 00 of card #1 and
                           channel 02 of card #2.
```

[ROUTe:]CLOSe?

[ROUTe:]CLOSe? *<channel_list>* returns the current state of the channel(s) queried. The *channel_list* is in the form (@ccgs). The command returns "1" if the channel is closed or returns "0" if the channel is open.

Comments

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

***Channel_list* Definition:** See [ROUTe:]CLOSe for *channel_list* definition.

Example Querying Channel Closure State

This example closes channels 100 and 202 of a two-module switchbox (card numbers 01 and 02) and queries channel closure. Since the channels are programmed to be closed, "1" is returned.

```
CLOS  (@100,202)         ! Close channel 00 of card #1 and
                           channel 02 of card #2.
CLOS?  (@202)            ! Query channel 02 of card #2.
```

[ROUTe:]OPEN

[ROUTe:]OPEN *<channel_list>* opens the Microwave Switch channels specified in the *channel_list*. The *channel_list* is in the form of (@ccgs), where cc = card number (01-99) and gs = channel number (microwave switch type dependant).

Parameters

Name	Type	Range of Values	Items
<*channel_list*>	numeric	1 - 99	card number (cc)
	numeric	g = 0 - 5, s = 0 - 3 (for 87104A,B,C) g = 0 - 5, s = 0 - 5 (for 87106A,B,C)	channel number (gs)

Comments

Opening Channels: To open:

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

NOTE *Opening order for multiple channels with a single command is not guaranteed. A list of channels will not all open simultaneously. Use sequential OPEN commands if needed.*

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

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

Example

Opening Microwave Switch Channels

This example opens channel 00 of card #1 and channel 02 of card #2 in a two-module switchbox.

```
OPEN (@100,202)                    ! Open channels 100 and 202.
```

[ROUTe:]OPEN?

[ROUTe:]OPEN? <*channel_list*> returns the current state of the channel(s) queried. The *channel_list* is in the form (@ccgs). The command returns "1" if the channel is open or "0" if the channel is closed.

Comments

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

***Channel_list* Definition:** See [ROUTe:]OPEN command on page 74 for *channel_list* definition.

Example

Querying Channel Open State

This example opens channels 100 and 202 of a two-module switchbox (card numbers 01 and 02) and queries channel state. Since the channels are programmed to be open, "1" is returned.

```
OPEN  (@100,202)                   ! Open channels 100 and 202.
OPEN? (@202)                       ! Query channel 202.
```

[ROUTe:]SCAN

[ROUTe:]SCAN *<channel_list>* defines the channels to be scanned. The *channel_list* is in the form (@ccgs), where cc = card number (01-99) and gs = channel number (microwave switch type dependant).

Parameters

Name	Type	Range of Values	Items
<*channel_list*>	numeric	1 - 99	card number (cc)
	numeric	g = 0 - 5, s = 0 - 3 (for 87104A,B,C) g = 0 - 5, s = 0 - 5 (for 87106A,B,C)	channel number (gs)

Comments

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

Scanning Channels: To scan:

-- a single channel, use [ROUTe:]SCAN (@ccgs);
-- multiple channels, use [ROUTe:]SCAN (@ccgs,ccgs,...);
-- sequential channels, use [ROUTe:]SCAN (@ccgs:ccgs);
-- groups of sequential channels, use [ROUTe:]SCAN (@ccgs:ccgs, ccgs:ccgs);
-- or any combination of above.

Scanning Operation: When a valid channel list is defined, INITiate[:IMMediate] begins the scan and closes the first channel in the *channel_list*. Successive triggers from the source specified by TRIGger:SOURce advance the scan through the channel list.

Stopping Scan: See the ABORt command on page 56.

Related Commands: TRIGger, TRIGger:SOURce

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

Example

Scanning Channels Using External Triggers

This example uses external triggering (TRIG:SOUR EXT) to scan channels 100 through 103 of a single-module switchbox. The trigger source to advance the scan is the input to the "Trig In" on the E1406A command module. When INIT is executed, the scan is started and channel 100 is closed. Then, each trigger received at the "Trig In" port advances the scan to the next channel.

```
TRIG:SOUR  EXT                  ! Set trigger source to EXTernal.
SCAN  (@100:103)                ! Set channel list to be scanned.
INIT                            ! Start scanning cycle and close
                                  channel 100.
(trigger externally)            ! Advance scan to next channel.
```

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

Subsystem Syntax

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

The STATus system contains four registers (that is, they reside in a SCPI driver, not in the hardware), two of which are under IEEE 488.2 control: the Standard Event Register (*ESE?) and the Status Byte Register (*STB?). The operational status bit (OPR), service request bit (RQS), standard event summary bit (ESB), message available bit (MAV) and questionable data bit (QUE) in the Status Byte Register (bits 7, 6, 5, 4 and 3 respectively) can be queried with the *STB? command. Use the *ESE? command to query the "unmask" value for the Standard Event Register (the bits you want logically OR'd into the summary bit). Only bit 8 in the Operation Status Register is used by the E8483A module. This bit is called the scan complete bit which is set whenever a scan operation completes. You can find bit 8 set with the STAT:OPER:EVEN? query command. Executing STAT:OPER:ENABle 256 command allows only bit 8 to generate a summary bit (the decimal value for bit 8 is 256). The registers are set and queried using decimal weighted bit values. The decimal equivalents are shown in "Figure 4-1. Status System Register Diagram" on page 78.

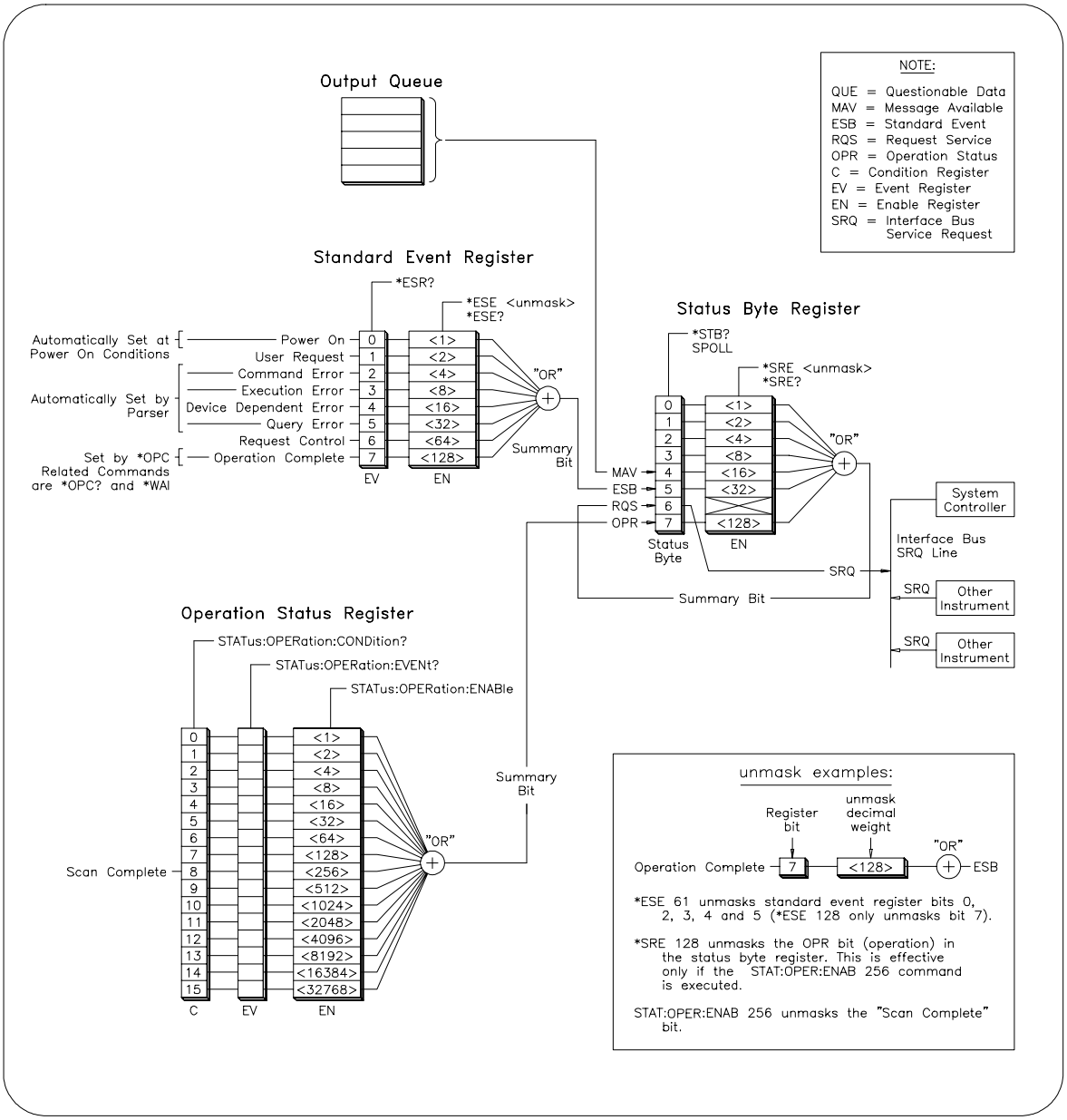


Figure 4-1. Status System Register Diagram

STATus:OPERation:ENABle

STATus:OPERation:ENABle *<unmask>* sets an enable mask to allow events recorded in the Event Register (Operation Status Group) to send a summary bit to the Status Byte Register (bit 7). For the E8483A module, when bit 8 in the Operation Status Register is set to 1 and that bit is enabled by the STATus:OPERation:ENABle command, bit 7 in the Status Byte Register is set to 1.

Parameters

Name	Type	Range of Values	Default Value
<*unmask*>	numeric	0 - 32767	N/A

Comments

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

Related Commands: [ROUTe:]SCAN

Example

Enabling Operation Status Register Bit 8

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

STATus:OPERation:ENABle?

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

Comments

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

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

Example

Querying the Operation Status Enable Register

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

STATus:OPERation[:EVENt]?

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

Comments

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

Returned Data after sending the STATus:OPERation[:EVENt]? Command: The command returns "+256" if bit 8 of the Operation Status Register is set to 1. The command returns "+0" if bit 8 of the Operation Status Register is set to 0.

Event Register Cleared: Reading the Event Register with the STATus:OPERation:EVENt? command clears it.

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

Related Commands: [ROUTe:]SCAN

Example

Reading the Operation Status Register After a Scanning Cycle

```
STAT:OPER?                  ! Return the bit values of the Operation
                              Status Register. Returns"+256" if
                              bit 8 is set to 1; "+0" if bit 8
                              is set to 0.
```

STATus:PRESet

STATus:PRESet affects only the Enable Register by setting all Enable Register bits to 0. It does not affect either the Status Byte Register or the Standard Event Status Register. PRESet does not clear any of the Event Registers.

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

Subsystem Syntax

```
SYSTem
    :CDEScription? <card_num>
    :COPTion <card_num>, <type>
    :COPTion? <card_num>
    :CPON <card_num> | ALL
    :CTYPe? <card_num>
    :ERRor?
    :VERSion?
```

SYSTem:CDEScription?

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

Parameters

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

Comments

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

```
"Six-Group Microwave Switch/Attenuator Driver"
```

Example

Reading the Description of Module #1

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

SYSTem:COPTion

SYSTem:COPTion *<card_num>, <type>* is used to set the model numbers of the microwave switches/attenuators being connected to the selected module.

Parameters

Name	Type	Range of Values	Default Value
<*card_num*>	numeric	1 - 99	N/A
<*type*>	string	Any six combination of AGT87104, AGT87106, AGT84904, AGT84906, AGT84907, UNKNOWN	N/A

Comments

The model numbers of the microwave switches and attenuators must be set according to the actual devices being connected. In addition, the "SW/ATN Identifier" switch provided on the module also requires to be set correctly. Refer to “Setting the SW/ATN Identifier Switch” on page 26 for more details.

The *<type>* parameter is a comma delineated list of model numbers of the microwave switches/attenuators being connected to the module’s connectors from Group 0 to Group 5. Valid model numbers can be AGT87104, or AGT87106, or AGT84904, or AGT84906, or AGT84907, or UNKNOWN as shown in Table 4-1. It can also be NULL which will leave the corresponding group setting unchanged.

For example, sending SYST:COPT 1, "AGT87106, , , , , " command will set the Group 0 to connect an Agilent 87106A/B/C Microwave Switch without changing other groups’ settings.

Table 4-1. *<type>* Parameter Description

Settings	Description
AGT87104	Indicates an Agilent 87104A, or 87104B, or 87104C Microwave Switch is being connected to the corresponding group connector.
AGT87106	Indicates an Agilent 87106A, or 87106B, or 87106C Microwave Switch is being connected to the corresponding group connector.
AGT84904	Indicates an Agilent 84904K, or 84904L Step Attenuator is being connected to the corresponding group connector.
AGT84906	Indicates an Agilent 84906K, or 84906L Step Attenuator is being connected to the corresponding group connector.
AGT84907	Indicates an Agilent 84907K, or 84907L Step Attenuator is being connected to the corresponding group connector.
UNKNOWN	Indicates no switch/attenuator, or a switch/attenuator other than the above type is being connected to the corresponding group connector.

Related Commands: SYST:COPT? *<card_num>*

NOTE *The <type> parameter must use capital letters.*

NOTE *Each time the settings are changed, you must recycle the E1406A command module to make the change effective.*

Example

Setting Models for the Connected Switches/Attenuators

Assuming two Agilent 87106A Microwave Switches are being connected to the Groups 0 & 1, two Agilent 84906K Step Attenuators to Groups 3 & 4. No devices are being connected to the Groups 2 & 5. Set the configuration by executing:

```
SYST:COPT  1, "AGT87106,AGT87106,UNKNOWN,AGT84906,AGT84906,UNKNOWN"
```

SYSTem:COPTion?

SYSTem:COPTion? *<card_num>* returns the model numbers of the microwave switches or attenuators set by the SYST:COPT command.

Parameters

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

Comments

The returned string is a comma delineated list of the model numbers of the switches/attenuators being connected to the module's connector from Group 0 to Group 5.

"UNKNOWN-SW" is returned if you set the switch type to "UNKNOWN". "UNKNOWN-ATN" is returned if you set the attenuator type to "UNKNOWN". Whether a switch or an attenuator being connected to the connector is determined by the SW/ATN Identifier Switch setting. See "Setting the SW/ATN Identifier Switch" on page 26 of this manual for details.

Related Commands: SYST:COPT <*card_num*>, <*type*>

Example

Querying Types of the Connected Switches/Attenuators

Assuming two Agilent 87106A Microwave Switches are being connected to the Groups 0 & 1, two Agilent 84906K Step Attenuators to Groups 3 & 4. No devices are being connected to the Groups 2 & 5. In addition, the Identifier Switch on the E8483A is set to the "Switch" position for Groups 0-2, and the "Attenuator" position for Groups 3-5.

```
SYST:COPT  1, "AGT87106, AGT87106, UNKNOWN, AGT84906, AGT84906, UNKNOWN"

SYST:COPT? 1                          ! "AGT87106, AGT87106, UNKNOWN-SW,
                                        AGT84906, AGT84906, UNKNOWN-ATN"
                                        will be returned.
```

SYSTem:CPON

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

Parameters

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

Comments

Differences between *RST and CPON: SYSTem:CPON ALL and *RST opens all channels of all modules in a switchbox, while SYSTem:CPON <*card_num*> opens the channels in only the module (card) specified in the command.

Example

Setting Card #1 Module to its Power-on State

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

SYSTem:CTYPe?

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

Parameters

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

Comments

Agilent E8483A Module Model Number: Sending this command returns:

```
HEWLETT-PACKARD,E8483A,<10-digit number>,A.11.01
```

where the <10-digit number> is the module's serial number 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? <*card_num*> *command is executed. Refer to DIAGnostic:TEST:SEEProm? command on page 62 for details.*

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

Example

Reading the Model Number of Module #1

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

SYSTem:ERRor?

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

Comments

Error Numbers/Messages in the Error Queue: Each error generated by a module stores an error number and corresponding error message in the error queue. The error message can be up to 255 characters long, but typically is much shorter.

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

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

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.
```

The **TRIGger** subsystem controls the triggering operation of the modules in a switchbox.

NOTE *This subsystem commands are used only for Microwave Switches during scanning.*

Subsystem Syntax

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

TRIGger[:IMMediate]

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

Comments

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

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

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

Example

Advancing Scan Using TRIGger Command

This example uses the TRIGger command to advance the scan of a single-module switchbox from channel 00 through 02. Since TRIGger:SOURce HOLD is set, the scan is advanced one channel each time TRIGger is executed.

```
TRIG:SOUR HOLD          ! Set trigger source to HOLD.
SCAN (@100:102)         ! Define channel list to be scanned.
INIT                    ! Start scanning cycle, close channel 100.
loop statement          ! Start count loop.
TRIG                    ! Advance scan to next channel.
increment loop          ! Increment loop count.
```

TRIGger:SOURce

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

Parameters

Name	Type	Parameter Description
BUS	discrete	*TRG or GET command
ECLTrg*n*	numeric	ECL Trigger bus line 0 - 1
EXTernal	discrete	"Trig In" port
HOLD	discrete	Hold Triggering
IMMediate	discrete	Immediate Triggering
TTLTrg*n*	numeric	TTL Trigger bus line 0 - 7

Comments

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

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

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

One Trigger Input Selected at a Time: Only one input (ECLTrg0 or 1; TTLTrg0, 1, 2, 3, 4, 5, 6 or 7; or EXTernal) can be selected at one time. Enabling a different trigger source will automatically disable the active input. For example, if TTLTrg1 is the active input, and TTLTrg4 is enabled, TTLTrg1 will become disabled and TTLTrg4 will become the active input.

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

Using External Trigger Inputs: With TRIGger:SOURce EXTernal selected, only one switchbox at a time can use the external trigger input at the E1406A "Trig In" port. The trigger input is assigned to the first switchbox requesting the external trigger source (with a TRIGger:SOURce EXTernal command).

Assigning EXTernal, TTLTrg*n*, and ECLTrg*n* Trigger Inputs: After using TRIGger:SOURce EXT|TTLT*n*|ECLT*n*, the selected trigger source remains assigned to the "SWITCH" driver until it is relinquished through use of the TRIG:SOUR BUS|HOLD command. While the trigger is in use by the "SWITCH" driver, no other drivers operating on the E1406 command module will have access to that particular trigger source. Likewise, other drivers may consume trigger resources which may deny access to a particular trigger by the "SWITCH" driver.

"Trig Out" Port Shared by Switchboxes: See the “OUTPut” on page 69 for more information.

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

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

Example Scanning Using External Triggers

This example uses external triggering (TRIG:SOUR EXT) to scan channels 00 through 03 of a single-module switchbox. The trigger source to advance the scan is the input applied to the Agilent E1406 command module's "Trig In" port. When INIT is executed, the scan is started and channel 00 is closed. Then each trigger received at the "Trig In" port advances the scan to the next channel.

```
TRIG:SOUR  EXT              ! Set trigger source to external.
SCAN  (@100:103)            ! Set channel list to be scanned.
INIT                        ! Start scanning cycle, close channel 100.
(trigger externally)        ! Advance channel list to next channel.
```

Example Scanning Using Bus Triggers

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

```
TRIG:SOUR  BUS              ! Set interface (BUS) triggering.
SCAN  (@100:102)            ! Set channel list to be scanned.
INIT                        ! Start scanning cycle, close channel 100.
loop statement              ! Loop to scan the channels.
*TRG                        ! Advance channel list to next channel.
(increment loop)            ! Increment loop count.
```

TRIGger:SOURce?

TRIGger:SOURce? returns the current trigger source for the switchbox. Command returns BUS, EXT, HOLD, IMM, TTLT0-7, or ECLT0-1 for sources BUS, EXTernal, HOLD, IMMediate, TTLTrg*n*, or ECLTrg*n*, respectively.

Example Querying the Trigger Source

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

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

SCPI Command Quick Reference

The following table summarizes the SCPI commands for the E8483A module. However, some subsystem commands are used only for controlling Microwave Switches and some only for controlling Step Attenuators. See each subsystem commands described in this chapter for details. Unless specially specified, all commands are applicable for both devices.

Command [a]		Description
ABORt [*]		Abort a scan in progress.
ARM [*]	:COUNt <*number*> :COUNt? [MIN|MAX]	Multiple scans per INIT command. Query number of scans.
ATTenuator [**]	:SET <*card_num*>,<*group_num*>,<*db*> :SET? <*card_num*>,<*group_num*>	Set attenuation value for the specified attenuator. Query the attenuation value set for the specified attenuator.
DIAGnostic [b]	:INTerrupt[:LINe] <*card_num*>,<*line_num*> :INTerrupt[:LINe]? <*card_num*> :INTerrupt:TIMer <*card_num*>,<*time*> :INTerrupt:TIMer? <*card_num*> :SCAN:DELay <*card_num*>,<*time*> :SCAN:DELay? <*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. Set additional scan delay time. Query the scan delay time. 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.
INITiate [*]	:CONTinuous <*mode*> :CONTinuous? [:IMMediate]	Enables/disables continuous scanning. Query continuous scan state. Starts a scanning cycle.
OUTPut [*]	:ECLTrg*n*[:STATe] ON |OFF | 1 | 0 :ECLTrg*n*[:STATe]? [:EXTernal][:STATe] ON | OFF | 1 | 0 [:EXTernal][:STATe]? :TTLTrg*n*[:STATe] ON |OFF | 1 | 0 :TTLTrg*n*[:STATe]?	Enables/disables the specified ECL trigger line. Query the specified ECL trigger line. Enables/disables the "Trig Out" port on the command module. Query the external state. Enables/disables the specified TTL trigger line. Query the specified TTL trigger line.
[ROUTe:] [*]	CLOSe <*channel _list*> CLOSe? <*channel _list*> OPEN <*channel_list*> OPEN? <*channel _list*> SCAN <*channel_list*>	Close channel(s). Query channel(s) closed. Open channel(s). Query channel(s) opened. Define channels for scanning.
STATus	:OPERation:CONDition? :OPERation:ENABle <*unmask*> :OPERation:ENABle? :OPERation[:EVENt]? :PRESet	Returns contents of the Operation Condition Register. Enables events in the Operation Event Register to be reported. Returns the unmask value set by the :ENABle command. Returns the contents of the Operation Event Register Sets Enable Register bits to 0.
SYSTem	:CDEScription? <*card_num*> :COPTion <*card_num*>, <*type*> :COPTion? <*card_num*> :CPON <*card_num*> | ALL :CTYPe? <*card_num*> :ERRor? :VERSion?	Returns description of module. Set the switch/attenuator types of the six groups. Query the switch/attenuator type set for the groups. Opens all channels on specified module(s) Returns the module type Returns error number/message in a module error queue. Returns the version of the SCPI standard.

Command [a]		Description
TRIGger [*]	[:IMMediate] :SOURce <*source*> :SOURce?	Causes a trigger event to occur. Set trigger source to BUS, or EXT, or HOLD, or IMM, or ECLTrgn, or TTLTrgn. Query scan trigger source.

a. The subsystem commands labeled with "[*]" in this table are used only for controlling microwave switches. The subsystem commands labeled with "[**]" in this table are used only for controlling attenuators.

b. All commands in the DIAGnostic subsystem are applicable for both microwave switches and attenuators, except the SCAN:DEL and the SCAN:DEL? commands are used for microwave switches only.

IEEE 488.2 Common Command Reference

The following table lists the IEEE 488.2 Common (*) Commands that can be accepted by the E8483A module.

Command	Command Description
*CLS	Clears all status registers (see STATus:OPERation[:EVENt]?) and clears the error queue.
*ESE *<register value>*	Enable Standard Event.
*ESE?	Enable Standard Event Query.
*ESR?	Standard Event Register Query.
*IDN?	Instrument ID Query; returns identification string of the module.
*OPC	Operation Complete.
*OPC?	Operation Complete Query.
*RCL *<numeric state>*	Recalls the instrument state saved by *SAV. You must reconfigure the scan list.
*RST	Resets the module. Opens all channels and invalidates current channel list for scanning. Sets ARM:COUN 1, TRIG:SOUR IMM, and INIT:CONT OFF.
*SAV *<numeric state>*	Stores the instrument state but does not save the scan list.
*SRE *<register value>*	Service request enable, enables status register bits.
*SRE?	Service request enable query.
*STB?	Read status byte query.
*TRG	Triggers the module to advance the scan when scan is enabled and trigger source is TRIGger:SOURce BUS.
*TST?	Self-test. Executes an internal self-test and returns only the first error encountered. Does not return multiple errors. The following is a list of responses you can obtain where "cc" is the card number with the leading zero deleted. +0 if self test passes. +cc01 for firmware error. +cc02 for bus error (problem communicating with the module). +cc03 for incorrect ID information read back from the module's ID register. +cc05 for hardware and firmware have different values. Possibly a hardware fault or an outside entity is register programming the E8483A. +cc10 if an interrupt was expected but not received. +cc11 if the busy bit was not held for a sufficient amount of time.
*WAI	Wait to Complete.

Notes:

Appendix A
E8483A Specifications

ITEMS	SPECIFICATIONS
Drive Output Per Switch [a]:	+5 V or +24 V
Maximum Energizing Voltage [a]:	42 V ac peak
Power Requirements:	Peak Module Current: 0.1 A @ +5 V or 1.2 A @ +24 V Dynamic Module Current [b]: 0.1 A @ +5 V or 1.2 A @ +24 V
Watts/slot:	15 W
Cooling/slot [b]:	0.08 mm H_2O @ 1.2 Liter/sec for 10^oC rise
Module Size/Device Type:	C-Size 2-Slot, Register based, A16, slave only, P1 and P2 Connectors
Operating Temperature:	0 - 55^oC
Operating Humidity:	65% RH, 0 - 40^oC

a. Control circuit can switch a maximum of 1 A per switch. Maximum current also depends on the output capability of the mainframe used.
b. Power and cooling requirements depend on switches installed or switches/attenuators connected.

Notes:

Appendix B
Register-Based Programming

About This Appendix

The Agilent E8483A Microwave Switch/Attenuator Driver 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 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 Agilent 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 96 shows the register address location within A16 as it might be mapped by an embedded controller. Figure B-2 on page 97 shows the location of A16 address space in the 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 module's logical address, and 64 (40_h) is the number of address bytes per register-based module. For example, the module's factory set logical address 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)*

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

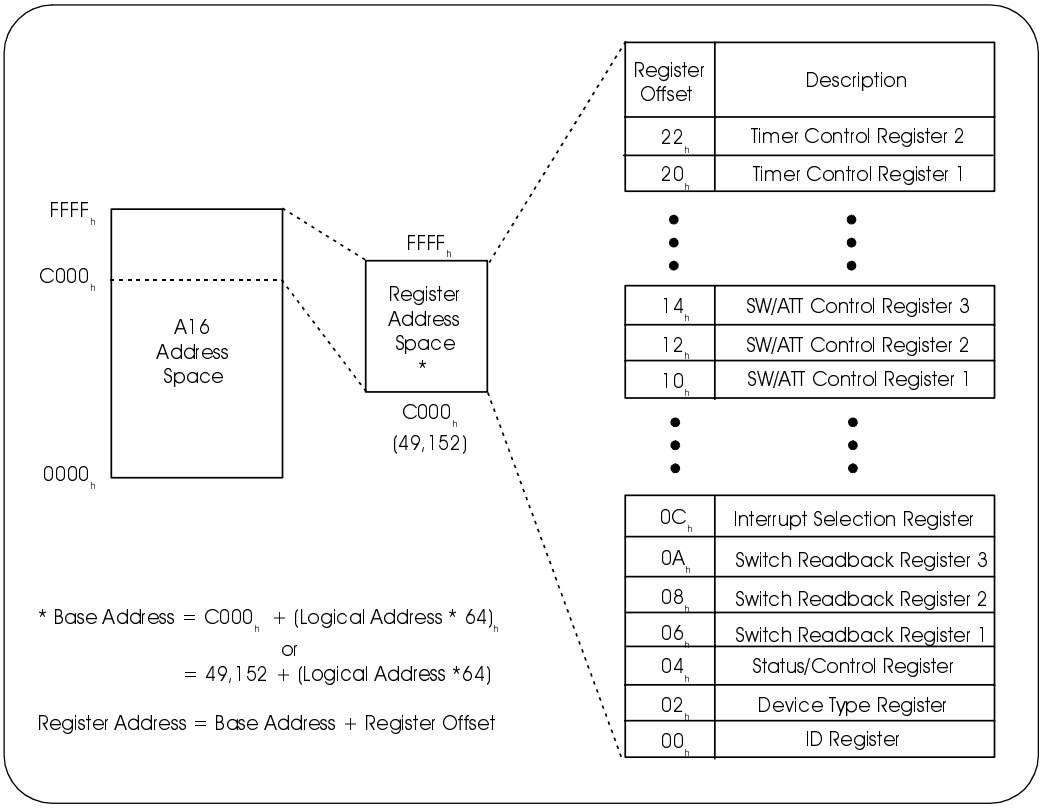


Figure B-1. Registers within A16 Address Space

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

When the A16 address space is inside the Agilent E1406A command module (Figure B-2), the multiplexer'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 multiplexer 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

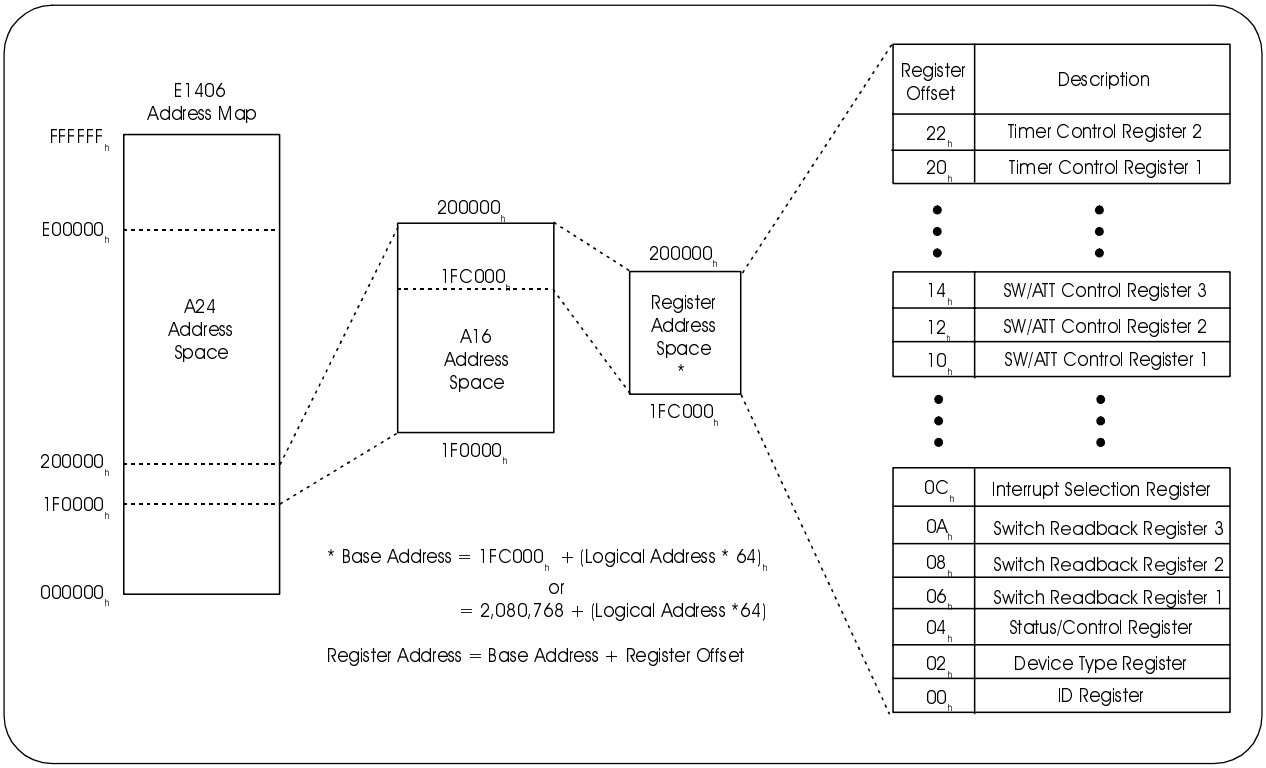


Figure B-2. Registers within Command Module's A16 Address Space

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 E8483A module contains twelve 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
Switch Readback Register 1 (for Group 0 and 1)	06_h	R	base + 06_h
Switch Readback Register 2 (for Group 2 and 3)	08_h	R	base + 08_h
Switch Readback Register 3 (for Group 4 and 5)	$0A_h$	R	base + $0A_h$
Interrupt Selection Register	$0C_h$	R/W	base + $0C_h$
Switch/Attenuator Control Register 1 (for Group 0 and 1)	10_h	R/W	base + 10_h
Switch/Attenuator Control Register 2 (for Group 2 and 3)	12_h	R/W	base + 12_h
Switch/Attenuator Control Register 3 (for Group 4 and 5)	14_h	R/W	base + 14_h
Timer Control Register 1	20_h	R/W	base + 20_h
Timer Control Register 2	22_h	R/W	base + 22_h

ID Register

The Manufacturer Identification Register is at offset address 00_h. Reading the register 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								Logical Address							
Read	Manufacturer ID - returns $FFFF_h$ in Agilent Technologies A16 only register-based card															

Device Type Register

The Device Type Register is at offset address 02_h. Reading the register returns $02D3_h$ indicating that the device is an E8483A module.

base + 02_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	x															
Read	$02D3_h$															

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 value 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 20 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 (all channels open).

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 by operating relays). 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.*

Switch Readback Registers

There are three Switch Readback registers used to read back the state of the microwave switches. These registers can not be written to.

-- Switch Readback Register 1 for Group 0 & 1 (Base + 06_h)
-- Switch Readback Register 2 for Group 2 & 3 (Base + 08_h)
-- Switch Readback Register 3 for Group 4 & 5 (Base + $0A_h$)

Switch Readback Register for Group 0 & 1 (base + 06_h)

base + 06_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Group 1								Group 0							
Read [a]	S/A	x	6	5	4	3	2	1	S/A	x	6	5	4	3	2	1

a. RF paths 1& 4 are not available for the Agilent 87104A/B/C series of microwave switch(es), so bits 0&3 for Group 0 and bits 8&11 for Group 1 are not used when the group(s) are used to control the 87104A/B/C switch(es).

Switch Readback Register for Group 2 & 3 (base + 08_h)

base + 08_h	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read [a]	Group 3								Group 2							
	S/A	x	6	5	4	3	2	1	S/A	x	6	5	4	3	2	1

a. RF paths 1& 4 are not available for the Agilent 87104A/B/C series of microwave switch(es), so bits 0&3 for Group 2 and bits 8&11 for Group 3 are not used when the group(s) are used to control the 87104A/B/C switch(es).

Switch Readback Register for Group 4 & 5 (base + $0A_h$)

base + $0A_h$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read [a]	Group 5								Group 4							
	S/A	x	6	5	4	3	2	1	S/A	x	6	5	4	3	2	1

a. RF paths 1& 4 are not available for the Agilent 87104A/B/C series of microwave switch(es), so bits 0&3 for Group 4 and bits 8&11 for Group 5 are not used when the group(s) are used to control the 87104A/B/C switch(es).

The numbers shown in the above register maps indicate the RF path numbers of the microwave switch.

- Reading the Switch Readback Registers returns a hexadecimal number. A bit that is "0" represents the related RF path is closed. A bit that is "1" indicates the related RF path is open.
- The highest bit of each group (bit 7 for Groups 0, 2, 4, and bit 15 for Groups 1, 3, 5) is used to identify whether a microwave switch or an attenuator is being controlled by the specified group. "1" indicates a switch and "0" indicates an attenuator.

NOTE *Reading the channel bit indicates to get the state of the relay driver circuit only. It cannot detect a defective relay.*

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 of the module.

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 equal to interrupt level 1 through 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 *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.*

Switch/Attenuator Control Registers

There are three Switch/Attenuator Control registers used to control the microwave switches and step attenuators. They are:

-- Switch/Attenuator Control Register 1 for Group 0 & 1 (Base + 10_h)
-- Switch/Attenuator Control Register 2 for Group 2 & 3 (Base + 12_h)
-- Switch/Attenuator Control Register 3 for Group 4 & 5 ((Base +14_h)

Switch/Attenuator Control Register for Group 0 & 1 (base + 10_h)

base + 10_h		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Group 1								Group 0							
Write	Switch [a]	OA	x	6	5	4	3	2	1	OA	x	6	5	4	3	2	1
	Attenuator[b]	Section 4		Section 3		Section 2		Section 1		Section 4		Section 3		Section 2		Section 1	
Read		Always Returns $FFFF_h$															

a. RF paths 1&4 are not available for the Agilent 87104A/B/C series of microwave switch(es), so bits 0&3 for Group 0 and bits 8&11 for Group 1 are not used when the group(s) are used to control the 87104A/B/C switch(es).

b. Section 4 is not available for the Agilent 84907K/L Step Attenuator, so bits 6&7 for Group 0 and bits 14&15 for Group 1 are not used when the group(s) are used to control the 84907K/L attenuator(s).

Switch/Attenuator Control Register for Group 2 & 3 (base + 12_h)

base + 12_h		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Group 3								Group 2							
Write	Switch [a]	OA	x	6	5	4	3	2	1	OA	x	6	5	4	3	2	1
	Attenuator[b]	Section 4		Section 3		Section 2		Section 1		Section 4		Section 3		Section 2		Section 1	
Read		Always Returns $FFFF_h$															

a. RF paths 1&4 are not available for the Agilent 87104A/B/C series of microwave switch(es), so bits 0&3 for Group 2 and bits 8&11 for Group 3 are not used when the group(s) are used to control the 87104A/B/C switch(es).

b. Section 4 is not available for the Agilent 84907K/L Step Attenuator, so bits 6&7 for Group 2 and bits 14&15 for Group 3 are not used when the group(s) are used to control the 84907K/L attenuator(s).

Switch/Attenuator Control Register for Group 4 & 5 (base + 14_h)

base + 14_h		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Group 5								Group 4							
Write	Switch [a]	OA	x	6	5	4	3	2	1	OA	x	6	5	4	3	2	1
	Attenuator[b]	Section 4		Section 3		Section 2		Section 1		Section 4		Section 3		Section 2		Section 1	
Read		Always Returns $FFFF_h$															

a. RF paths 1&4 are not available for the Agilent 87104A/B/C series of microwave switch(es), so bits 0&3 for Group 4 and bits 8&11 for Group 5 are not used when the group(s) are used to control the 87104A/B/C switch(es).

b. Section 4 is not available for the Agilent 84907K/L Step Attenuator, so bits 6&7 for Group 4 and bits 14&15 for Group 5 are not used when the group(s) are used to control the 84907K/L attenuator(s).

The bit definitions of these registers for controlling switches are different from the bit definitions for controlling attenuators.

To Control Switches

When these registers are used to control switches, the numbers shown in the above register maps indicate the RF path numbers to be controlled.

- Writing a "1" to one bit will close the related RF path, and writing a "0" will open the RF path.
- Writing a "1" to bit 7 (OA) of these registers will open all paths of the related switches (groups 0, 2, and 4). Writing a "1" to bit 15 (OA) of these registers will open all paths of the related switches (groups 1, 3, and 5).

For example, to close RF path 2 of the Group 0 switch, write a "1" to bit 1 of the Switch/Attenuator Control Register (base $+10_h$). To close all RF paths of Group 3 switch, write a "1" to bit 15 of the Switch/Attenuator Control Register (base $+12_h$).

To Control Attenuators

When these registers are used for attenuators, the numbers shown in the above register maps indicate the section number of each attenuator.

- Each section uses two bits, the lower bit is for driving Thru Line pin of the attenuator and the higher bit for Attn Card pin.
- To set a section attenuation value to 0 dB, write a "1" to the Thru Line pin of the section. To set a section attenuation to the specified value (dependant upon the type of the attenuator), write a "1" to the Attn Card pin of the section.

Table B-2 shows the section attenuation value of the attenuators. Table B-3 shows the recommended switching sequence for the attenuators. For more information about the listed attenuators, refer to the corresponding Technical Data Sheet.

Table B-2. Solenoid Pin and Attenuation Guide

Attenuator Type	Section 1		Section 2		Section 3		Section 4	
	Thru Line	Attn Card	Thru Line	Attn Card	Thru Line	Attn Card	Thru Line	Attn Card
84904K/L	0	1 dB	0	2 dB	0	4	0	4
84906K/L	0	10	0	20	0	30	0	30
84907K/L	0	10	0	20	0	40	X	X

Table B-3. Recommended Attenuation Guide

Attenuator Type	Attenuation Value (dB)											
84907K/L	0	10	20	30	40	50	60	70				
Section 1 (10 dB) Section 2 (20 dB) Section 3 (40 dB)		x	 x	x x	 x	x x	 x x	x x x				
84906K/L	0	10	20	30	40	50	60	70	80	90		
Section 1 (10 dB) Section 2 (20 dB) Section 3 (30 dB) Section 4 (30 dB)		x	 x	 x	x x	 x x	 x x	x x x	 x x x	x x x x		
84904K/L	0	1	2	3	4	5	6	7	8	9	10	11
Section 1 (1 dB) Section 2 (2 dB) Section 3 (4 dB) Section 4 (4 dB)		x	 x	x x	 x	x x	 x x	x x x	 x x	x x x	 x x x	x x x x

For example, to set Agilent 84906K Step Attenuator (which is connected to the 10-pin Group 0 connector of the E8483A module) to 30 dB, write a "1" to bits 0, 2, 5, and 6 of the Switch/Attenuator Control Register (base +10_h). To set the attenuation value to 70 dB, write a "1" to bits 1, 2, 5 and 7 of the register (base +10_h).

Timer Control Registers

There are two registers used to provide a programmable timer for the relay settling time. The settling time for the E8483A module depends on what switches/attenuators are being controlled by the module.

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

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

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	x								Read Time							

NOTE *For register-based programming, you must set the relay settling time each time the module is powered up.*

NOTE *We highly recommend you set the time no less than 15 ms for the 87104/6A, B, C series microwave switches and no less than 20 ms for 84904/6/7K,L series step attenuators. If different types of switches and attenuators are to be controlled by the module, set the settling time to the maximum of them.*

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 20 ms, you should write 1DFF_h to Timer Control Register 1 (base + 20_h) and FB_h to Timer Control Register 2 (base + 22_h). Since:

$$xxyyyy_h = \text{FFFFFF}_h - (20 \text{ ms} / 62.5 \text{ ns})_h = \text{FB1DFF}_h$$

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

Notes:

Appendix C
Error Messages

Table C-1 lists the error messages associated with the E8483A Microwave Switch/Attenuator Driver Module when programmed with SCPI commands. See the appropriate mainframe manual for a complete list of error messages.

Table C-1. Microwave Switch/Attenuator Driver Module Error Messages

Number	Error Message	Potential Cause(s)
-211	Trigger ignored	Trigger received when scan not enabled. Trigger received after scan complete. Trigger too fast.
-213	Init Ignored	Attempting to execute an INIT command when a scan is already in progress.
-224	Illegal parameter value	Attempting to execute a command with a parameter not applicable to the command.
-240	Hardware error	Command failed due to a hardware problem.
-310	System error	Internal driver error. This error can result if an excessively long parameter list is entered.
1500	External trigger source already allocated	Assigning an external trigger source to a switchbox when the trigger source has already been assigned to another switchbox.
2000	Invalid card number	Addressing a module (card) in a switchbox that is not part of the switchbox.
2001	Invalid channel number	Attempting to address a channel of a module in a switchbox that is not supported by the module (e.g., channel 99 of a module).
2006	Command not supported on this card	Sending a command to a module (card) in a switchbox that is unsupported by the module.
2008	Scan list not initialized	Executing a scan without the INIT command.
2009	Too many channels in channel list	Attempting to address more channels than available in the switchbox.
2012	Invalid Channel Range	Invalid channel(s) specified in SCAN <*channel_list*> command. Attempting to begin scanning when no valid channel list is defined.
2600	Function not supported on this card	Sending a command to a module (card) in a switchbox that is not supported by the module or switchbox.
2601	Missing parameter	Sending a command requiring a *channel_list* without the *channel_list*.

Notes:

A

B

C

C (*continued*)

D

E

F

G

H

I

Q

R

S

S (*continued*)

T

V

W

Notes:

Agilent Technologies

Manual Part Number: E8483-90001
Printed in U.S.A. E0301