Agilent E1339A/Z2309A 72-Channel Open Collector Digital Output/Relay Driver Module

User's and SCPI Programming Manual

Where to Find it - Online and Printed Information:

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

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

Agilent Technologies

Manual Part Number: E1339-90001
Printed in Malaysia E0912

Contents

Warranty 7
Safety Symbols 8
WARNINGS 8
Declaration of Conformity 9-10
User's Notes 11-12

Chapter 1
Getting Started 13
Using this Chapter 13
Relay Driver Module Description 13
Relay Driver Module Operation 14
Modes of Operation 15
Using 72-Channel Mode 16
Using 36-Channel Mode 16
Relay Driver Module Programming 17
Switchbox Definition 17
Specifying SCPI Commands 18

Chapter 2
Configuring the Relay Driver Module 21
Using this Chapter 21
Warnings and Cautions 21
Relay Driver Module Configuration 22
Connecting User Inputs 22
Setting Interrupt Priority Jumpers 24
Setting Logical Address Switch 25
Setting Internal Power Supply Voltage 25
Setting Relay Driver Module Operation 26
Operations Overview 26
Setting 36- or 72-Channel Mode 26
Setting Simultaneous/Stepped Actuation (36-Channel Mode) 27
Setting Pulsed/Continuous Output (36-Channel Mode) 27
Changing Timer Delay 27
Installing the Relay Driver Module 28
Hardware Installation 28
Software Installation 28
Verifying Initial Operation 29

Chapter 3
Using the Relay Driver Module 31
Using This Chapter 31
Using 72-Channel Mode 32
Operations Overview 32
Switching Channels 32
Scanning Channels 32
Using 36-Channel Mode 33
Operations Overview 33

Setting Simultaneous/Stepped Actuations 34
Setting Pulsed/Continuous Outputs 34
Switching Channel Pairs 34
Scanning Channel Pairs 35
Example Programs 37
Using the Example Programs 37
Example: Closing a Channel (72-Channel Mode) 38
Example: Closing a Channel (36-Channel Mode) 40
Example:Scanning Channels (72-Channel Mode) 42
Example: Scanning Channels (36-Channel Mode) 44

Chapter 4
Understanding the Relay Driver Module 47
Using This Chapter 47
Scanning Channels 47
Setting Scan Mode Operation 47
Using Scanning Trigger Sources 47
Scanning With External Instruments 50
Using the Scan Complete Bit 50
Setting/Checking States 50
Storing States (*SAV) 50
Recalling States (*RCL) 50
Response to Reset (*RST) 51
Error Query Response (SYST:ERR?) 51
Example Programs 51
Example: Scanning Using Trig Out /Event In Ports 51
Example: Scan Complete Interrupt 55

Chapter 5
Relay Driver Module Command Reference 57
Using This Chapter 57
Command Types 57
ABORt Subsystem 60
ARM Subsystem 62
ARM:COUNt <*number*> MIN | MAX 62
ARM:COUNt? [MIN | MAX] 63
DISPlay Subsystem 64
DISPlay:MONitor:CARD <*number*>| AUTO 64
DISPlay:MONitor:CARD? 64
DISPlay:MONitor[:STATe] <*mode*> 65
DISPlay:MONitor[:STATe]? 65
INITiate Subsystem 66
INITiate:CONTinuous <*mode*> 66
INITiate:CONTinuous? 67
INITiate[:IMMediate] 67

OUTPut Subsystem ... 69
OUTPut:ECLT*n*[:STATe] <*mode*> ... 69
OUTPut:ECLTn[:STATe]? ... 70
OUTPut[:EXTernal][:STATe] ... 70
OUTPut[:EXTernal][:STATe]? ... 71
OUTPut:TTLTrgn[:STATe] <*mode*> ... 72
OUTPut:TTLTrgn[:STATe]? ... 73
[ROUTe:] Subsystem ... 74
[ROUTe:]CLOSe <*channel_list*> ... 74
[ROUTe:]CLOSe? <*channel_list*> ... 75
[ROUTE:]OPEN <*channel_list*> ... 76
[ROUTe:]OPEN? <*channel_list*> ... 77
[ROUTE:]SCAN <*channel_list*> ... 77
STATus Subsystem ... 79
STATus:OPERation:CONDition? ... 81
STATus:OPERation:ENABle <*unmask*> ... 81
STATus:OPERation:ENABle? ... 81
STATus:OPERation[:EVENt]? ... 82
STATus:PRESet ... 82
SYSTem Subsystem ... 83
SYSTem:CDEScription? <*number*> ... 83
SYSTem:CPON <*number*> | ALL ... 83
SYSTem:CTYPE? <*number*> ... 84
SYSTem:ERRor? ... 85
TRIGger Subsystem ... 86
TRIGger[:IMMediate] ... 86
TRIGger:SOURce <*source*> ... 87
TRIGger:SOURce? ... 89
SCPI Command Quick Reference ... 90
IEEE 488.2 Common Commands ... 91

Appendix A
Relay Driver Specifications ... 93

Appendix B
Relay Driver Register Definitions ... 95
Types of Registers ... 95
Addressing the Registers ... 95
Register Access with Logical Address ... 96
Register Access with Memory Mapping ... 96
Reading from the Registers ... 97
ID Register ... 97
Device Type Register ... 97
Status/Control Register ... 97
Configuration Registers ... 98
Channel Enable Registers ... 98

Writing to the Registers 98
Status/Control Register 98
Channel Enable Registers 98
Register Definitions 99
ID Register 99
Device Type Register 99
Status/Control Register 99
Configuration Registers 100
Channel Enable Registers 101

Appendix C
Relay Driver Error Messages 103
Error Types 103
Error Messages 103

Certification

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

AGILENT TECHNOLOGIES WARRANTY STATEMENT

PRODUCT: E1339A/Z2309A **DURATION OF WARRANTY:** 1 year

1. Agilent warrants Agilent hardware, accessories and supplies against defects in materials and workmanship for the period specified above. If Aglent 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 Technologies, (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.

IEC Measurement Category II Overvoltage Protection

This is a measurement Category II product designed for measurements at voltages up to 300V from earth, including measurements of voltages at typical mains socket outlets. The product should not be used to make voltage measurements on a fixed electrical installation including building wiring, circuit breakers, or service panels.

E1339A/Z2309A 72-Channel Open Collector Digital Output/Relay Driver Module User's Manual

Agilent Technologies

Edition 2 Rev 3

Documentation History

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

Edition 1 (Part Number Z2309-90001) . . . . . September 1991

Edition 2 (Part Number E1339-90001) . . . . . April 1997

Edition 2 Rev 2 (Part Number E1339-90001) . . . . . May 2006

Edition 2 Rev 3 (Part Number E1339-90001) . September 2012

Trademarks

Microsoft® is a U.S. registered trademark of Microsoft Corporation

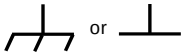
Windows NT® is a U.S. registered trademark of Microsoft Corporation

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation

Safety Symbols

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

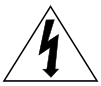
Indicates the field wiring terminal that must be connected to earth ground before operating the equipment—protects against electrical shock in case of fault.



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

Alternating current (AC)

Direct current (DC).



Indicates hazardous voltages.

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

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

WARNINGS

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

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

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

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

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

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

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

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

Declaration of Conformity

Declarations of Conformity for this product and for other Agilent products may be downloaded from the Internet. There are two methods to obtain the Declaration of Conformity:

- Go to http://regulations.corporate.agilent.com/DoC/search.htm . You can then search by product number to find the latest Declaration of Conformity.
- Alternately, you can go to the product web page (www.agilent.com/find/E1339A), click on the Document Library tab then scroll down until you find the Declaration of Conformity link.

Notes:

Notes:

Notes:

Chapter 1
Getting Started

Using this Chapter

This chapter shows how to get started using the Agilent E1339A / Z2309A 72-channel Open Collector Digital Output/Relay Driver module (called the Relay Driver Module in this manual). The chapter includes:

- Relay Driver Module Description. . . . . page 13
- Relay Driver Module Operation . . . . . page 14
- Relay Driver Module Programming . . . . . page 17

Relay Driver Module Description

The Relay Driver Module is a VXIbus and VMEbus B-Size register-based slave device. Since the Relay Driver can be used in a B-Size or C-Size VXIbus or VMEbus mainframe, the term **Command Module** is used in this manual to refer to the Agilent E1306 or Agilent E1406 Command Module or to Agilent E13XX mainframes.

As shown in Figure 1-1, the Relay Driver Module consists of 72 channels (channels 00 through 71). Each channel consists of an open collector output that is activated by turning on the channel driver, allowing current to be sunk. When the output is deactivated the driver is off, not allowing current to be sunk.

The Relay Driver Module can switch voltages up to 32V and sink up to 200 mA per channel. An internal +12V power supply is available to drive external loads. When used in a VXIbus mainframe with P2 capability, internal +5V and +24V power supplies are also available.

The Relay Driver Module also has 24 input bits that can be used for configuration. These bits are used with the SCPI driver to determine in which mode (36-channel or 72-channel) the card is to be used. A 30 msec timer is available for interrupts (IRQ) and for delays between channels.

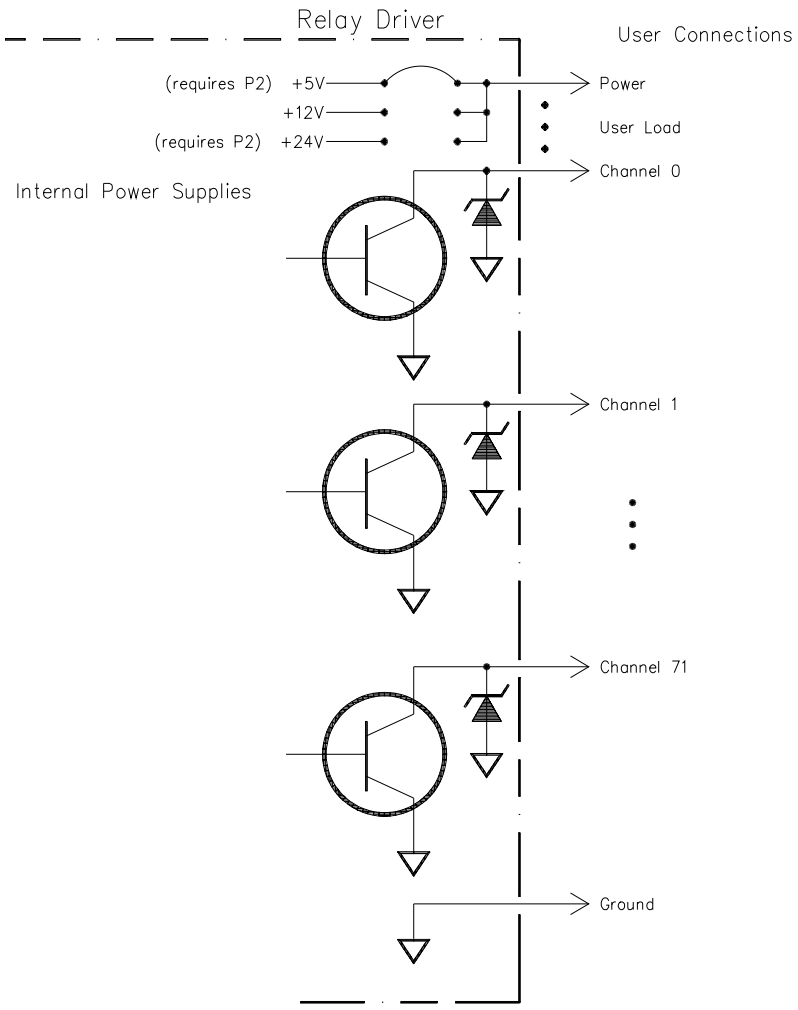


Figure 1-1. Relay Driver Module Schematic

Relay Driver Module Operation

For a SCPI environment, multiple Relay Driver Modules can be configured as a switchbox instrument. When the Agilent E1339A SCPI driver is downloaded, the switchbox can be comprised of one or more Agilent E1339A Relay Driver Modules, one or more Z2309 Relay Driver Modules, or a combination of Agilent E1339A and Z2309 Relay Driver Modules.

The Relay Driver Module can be configured for 36-channel or 72-channel mode. In 72-channel mode, each driver is independently controlled. In 36-channel mode two bits are used per channel, with one bit being the inverse of the other bit. This mode can be used to drive two coil switches, one for SET and one for RESET. For 36-channel mode, simultaneous actuation, stepped actuation, pulsed output, and/or continuous output operations are available.

Modes of Operation

Table 1-1 summarizes the modes of operation for the Relay Driver Module. For 36-channel operation, note that CNFG 1 sets the actuation mode (simultaneous actuation or stepped actuation) for the channel pairs, while CNFG 2 sets the method of current output from the channel pairs (continuous output or pulsed output).

Table 1-1. Relay Driver Module Operating Modes

Operation	Switching One Channel	Switching Multiple Channels	Scanning Channels
72-Channel Mode (Set when CNFG 0 is Open)			
	OPEN (@ccnn) opens the channel and CLOSe (@ccnn) closes the channel.	All channels in the OPEN/CLOSe *channel_list* are actuated nearly simultaneously.	A 60-msec delay occurs between actuations of each channel in the SCAN *channel_list*.
36-Channel Mode Actuation (Valid only when CNFG 0 is connected to GROUND)			
Simultaneous Actuation (CNFG 1 Open)	OPEN (@ccnn) opens the channel pair and CLOSe (@ccnn) closes the channel pair.	All channel pairs (SET/RESET) in the OPEN/CLOSe *channel_lis*t are actuated nearly simultaneously.	A 60-msec delay occurs between actuations of each channel pair in the SCAN *channel_list*.
Stepped Actuation (CNFG 1 connected to GROUND)	OPEN (@ccnn) opens the channel pair and CLOSe (@ccnn) closes the channel pair.	A 30-msec delay occurs between actuations of each channel pair in the OPEN/CLOSe *channel_list*.	A 60-msec delay occurs between actuations of each channel pair in the SCAN *channel_list*.
36-Channel Mode Current Output (Valid only when CNFG 0 is connected to GROUND)			
Continuous Output (CNFG 2 Open)	The SET or RESET output current is continuously applied to the channel pair when actuated.	The SET or RESET output current is continuously applied to the channel pair that is actuated.	The SET or RESET output current is continuously applied to the channel pair that is actuated.
Pulsed Output (CNFG 2 connected to GROUND)	When the channel pair is actuated, the SET or RESET output current is applied until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs.	The SET or RESET output current is applied to the actuated channel pair until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs This process is then repeated with the next channel pair in the OPEN/CLOSe *channel_list* until the list is completed.	The SET or RESET output current is applied to the actuated channel pair until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs. For scanning, pulsed output requires twice as much time as continuous output to complete the operation.

Using 72-Channel Mode

Set 72-channel mode by leaving CNFG 0 open. For 72-channel mode operation, use OPEN <*channel_list*> to open channel(s) and thus not sink current, and CLOSe <*channel_list*> to close channel(s) and sink current. Use SCAN <*channel_list*> to scan channels.

When switching channel(s) (using OPEN <*channel_list*> or CLOSe <*channel_list*>), drivers are actuated essentially simultaneously (less than 100 μsec) if more than one channel is specified in the channel list. When scanning channels (using SCAN <*channel_list*>), there is a 60 msec delay time between actuation of each channel in the SCAN *channel_list*. This time includes 30 msec to close the channel pair, and another 30 msec to open the channel pair before moving to the next channel pair.

Using 36-Channel Mode

Set 36-channel mode by connecting CNFG 0 to GROUND. For 36-channel mode, four operations are available: simultaneous or stepped actuations and pulsed or continuous outputs. The term "channel pair" refers to the SET and RESET parts of the channel, such as CH0 SET and CH0 RESET.

Simultaneous/ Stepped Actuations

For 36-channel mode, use the CNFG 1 bit to set simultaneous or stepped actuations. Leave CNFG 1 open to set simultaneous actuations or connect CNFG 1 to GROUND to set stepped actuations.

When switching channel pairs with **simultaneous actuation**, channel pairs (SET/ RESET) are actuated essentially simultaneously (less than 100 μsec) if more than one channel is specified in OPEN <*channel_list*> or CLOSe <*channel_list*>. However, for more than one channel in the channel list, actuation order is not guaranteed.

When switching channel pairs with **stepped actuation**, an OPEN or CLOSe command waits for the internal 30 msec timer to interrupt between each channel pair. For the OPEN and CLOSe commands, the timer indicates completion of a relay actuation (caused by the driver). The instrument will wait for the time to elapse (causing an IRQ) before the next command is executed. On a *RST, up to six relays may be actuated at a time.

When **scanning channel pairs** with SCAN <*channel_list*>, a 60 msec delay occurs between actuations of each channel pair in the channel list for either simultaneous or stepped actuation. This time includes 30 msec to close the channel pair, and another 30 msec to open the channel pair before moving to the next channel pair.

Pulsed/ Continuous Output

For 36-channel mode, use the CNFG 2 bit to set pulsed or continuous output. Leave CNFG 2 open to set continuous output or connect CNFG 2 to GROUND to set pulsed output. With **continuous output** (for both switching and scanning channels), when a channel pair is actuated the SET or RESET output current is continuously applied to the channel pair.

With **pulsed output** (for both switching and scanning channels), when a channel pair is actuated the SET or RESET current is applied until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs. Pulsed output requires twice as much time as continuous output to complete the operation. On a *RST, up to six relays may be actuated at a time with pulsed output.

Relay Driver Module Programming

The Relay Driver Module can be programmed using register access (see *Appendix B - Relay Driver Module Register Definitions*) or via the Command Module using Standard Commands for Programmable Instruments (SCPI).

Note Since the Relay Driver Module can be used in a B-Size or C-Size mainframe, the term Command Module as used in this manual refers to the Agilent E1306 or Agilent E1406 Command Module or to the B-Size mainframe.

If you use SCPI to program the Relay Driver Module, it is highly recommended that you do not use register programming since the SCPI driver maintains an image of the card state. If the card state is altered by using register writes, the driver will be unaware of these changes.

Switchbox Definition

For a SCPI environment, multiple Relay Driver Modules can be configured as a **switchbox** instrument. The term switchbox refers to a SCPI instrument consisting of one or more switchcards (Relay Driver Modules). The instrument itself is the firmware running in the Command Module.

For a switchbox instrument, all channels within the instrument can be addressed using a single interface address. Single, multiple, or continuous scanning of specified switch channels is available for any switchbox configuration. Logical addresses should start on a boundary of 8 for the command module. Also, logical addresses are sequential.

Example: Typical Switchbox Instrument

For example, Figure 1-2 shows a typical switchbox with three Relay Driver Modules. The three modules have sequential logical addresses of 120, 121,

and 122. Since switch #1 has the lowest logical address, it is card number 01.

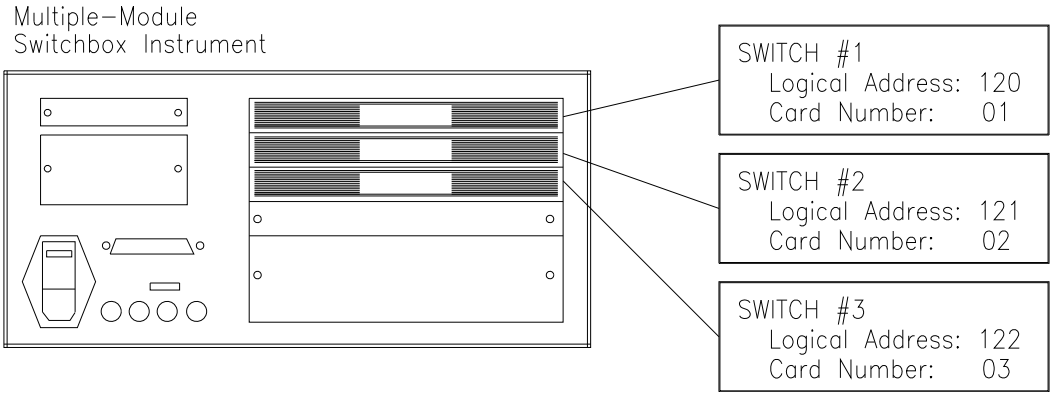


Figure 1-2. Example: Typical Switchbox Instrument

Specifying SCPI Commands

To program the Relay Driver Module using SCPI, you must select the computer language, interface address and SCPI commands to be used. Guidelines to select SCPI commands for the Relay Driver Module follow. See the *Agilent 75000 Series C (or B) Installation and Getting Started Guide* for interface addressing and computer language information.

To address channels within a Relay Driver Module, you must specify the SCPI command and switch channel address. For the Relay Driver Module module, use CLOSe <*channel_list*> to allow the driver to sink current. Use OPEN <*channel_list*> to stop the driver from sinking current. Use SCAN <*channel_list*> to close the set of channels specified, one channel at a time.

Switch Channel Addresses

For the Relay Driver Module, the channel address (*channel_list*) has the form (@ccnn) where cc = switch card number (01-99) and nn = channel numbers (00-71 or 00-35). You can address single channels (@ccnn); multiple channels (@ccnn,ccnn ...);sequential channels (@ccnn:ccnn); groups of sequential channels (@ccnn:ccnn,ccnn:ccnn); or any combination.

Switch Card Numbers

The switch card number depends on the switchbox configuration (single-module or multiple module) set for the switches. (Leading zeroes can be ignored for the card number.) For a single-module switchbox, the card number is always 01.

For a multiple-module switchbox, the card numbers are 01, 02,...,cc. The module with the lowest logical address is card number 01, the module with the next-lowest logical address is card number 02, etc. See the *Agilent 75000 Series C (or B) Installation and Getting Started Guide* for a definition of logical addresses.

Example: Three-Module Switchbox

For example, assume that three Relay Driver Modules are configured with

logical addresses of 120, 121, and 122. Card number 1 is assigned to the module at logical address 120, card number 2 is assigned to the module at logical address 121, and card number 3 is assigned to the module at logical address 122.

Switch Channel Numbers

Relay Driver Module channel numbers are 00 through 71 for 72-channel mode or 00 through 35 for 36-channel mode. The channels can be addressed using channel numbers or channel ranges. For a single-module switchbox, channel ranges can span across the channels. For multiple-module switchboxes, channel ranges can span across the channels of all modules.

Use commas (,) to form a channel list or use a colon (:) to form a channel range. Only valid channels can be accessed in a channel list or channel range, and the channel list or channel range must be from a lower channel number to a higher channel number. Some examples follow.

Example: Channel Lists/Ranges (72-Channel Mode)

For 72-channel mode, each channel is opened or closed independently using CLOSe to close a channel and OPEN ot open a channel.

Channel Lists:

```
CLOSe (@100,112)        ! Close chs 0 and 12 on card 1
OPEN (@203,210)         ! Open chs 3 and 10 on card 2
```

Channel Ranges:

```
OPEN (@100:171)         ! Open all channels on card 1
SCAN (@100:171)         ! Set scan list for all channels
                          on card 1
```

Example: Channel Lists/Ranges (36-Channel Mode)

For 36-channel mode, channel pairs (SET/RESET) are opened or closed using CLOSe to close a channel pair and OPEN to open a channel pair. When CLOSe is executed, the channel SET is closed and the channel RESET is opened. When OPEN is executed, the channel SET is opened and the channel RESET is closed.

Channel Lists:

```
CLOSe (@100,112)        ! Close chs 0 and 12 SET and
                          open chs 0 and 12 RESET
                          on card 1
OPEN (@203,210)         ! Open chs 3 and 10 SET and
                          close chs 3 and 10 RESET
                          on card 2
```

Channel Ranges:

```
OPEN (@100:135)         ! Open all channels SET and
                          close all chanels RESET on
                          card 1
SCAN (@100:135)         ! Set scan list for all channels
                          on card 1
```

Chapter 2
Configuring the Relay Driver Module

Using this Chapter

This chapter shows how to configure and install the Relay Driver Module. The chapter includes:

- Warnings and Cautions . . . . . page 21
- Relay Driver Module Configuration . . . . . page 22
- Setting Relay Driver Module Operation . . . . . page 26
- Installing the Relay Driver Module . . . . . page 28
- Verifying Initial Operation . . . . . page 29

Warnings and Cautions

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

WARNING **SHOCK HAZARD. When handling user wiring connected to the Relay Driver Module, consider the highest voltage present accessible on any channel. Use wire with an insulation rating greater than the highest voltage which will be present on any channel.**

Caution **MAXIMUM VOLTAGE/CURRENT. Maximum allowable voltage per channel for the Relay Driver Module module is 32 VDC and 200 ma per channel. Exceeding any limit may damage the module.**

STATIC-SENSITIVE DEVICE. Use anti-static procedures when removing, configuring, or installing a module.

Relay Driver Module Configuration

This section gives guidelines to configure the Relay Driver Module for installation, including:

- Connecting User Inputs
- Setting Interrupt Priority Jumpers
- Setting the Logical Address Switch
- Setting Internal Power Supply Level

Connecting User Inputs

User inputs to the Relay Driver Module are via 60-pin ribbon connectors to the P200 and P400 connectors on the faceplate as shown in Figure 2-1. To connect the ribbon cable to P200 or P400, push the ribbon connector pins onto the P200 or P400 pins until the side latches engage the ribbon cable connector.

You can also use the ribbon connector leads to set Relay Driver Module operations, as shown in "Setting Relay Driver Module Operation". Tables 2-1 through 2-4 show the pin definitions for the P200 and P400 connectors for 36-channel and 72-channel operation.

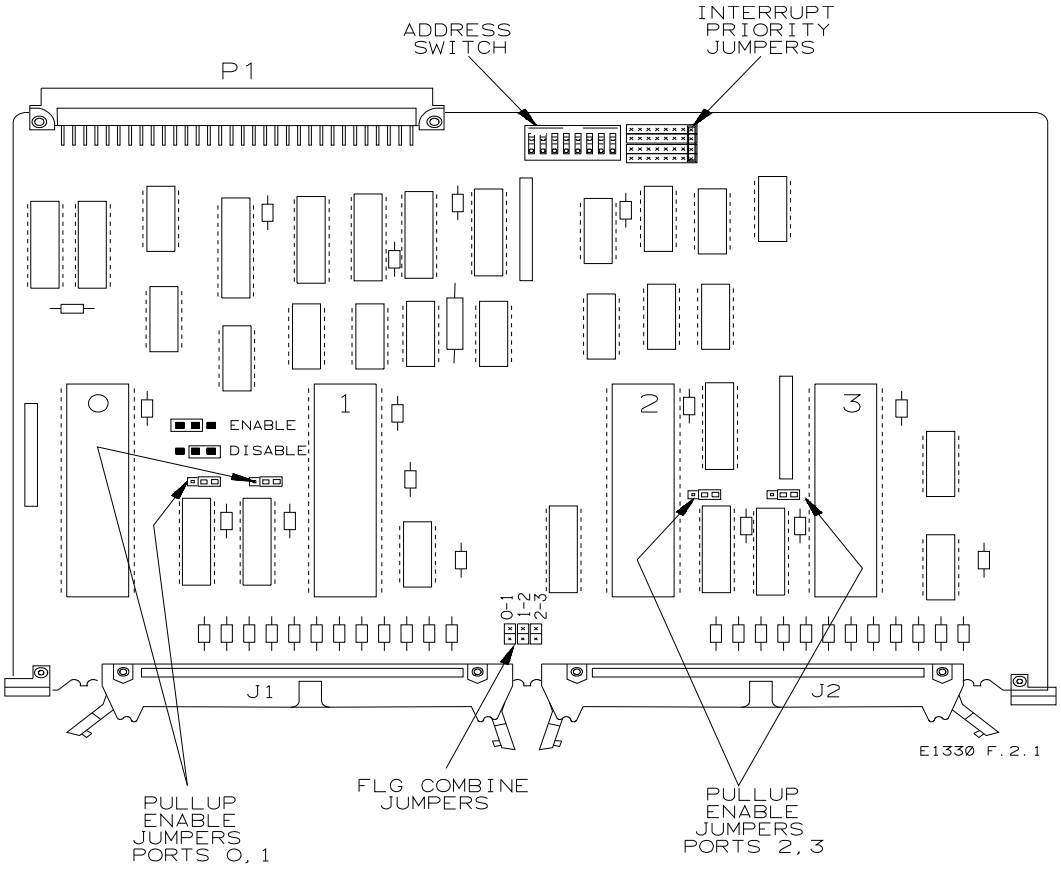


Figure 2-1. Connecting User Inputs

Table 2-1. 36-Channel Pin Definitions (P200)							
Pin #	Name	Pin #	Name	Pin#	Name	Pin#	Name
1	CH 18 S	16	CNFG 15	31	CH 4 S	46	CH 8 S
2	CH 18 R	17	CNFG 14	32	CH 4 R	47	CH 8 R
3	CH 19 S	18	CNFG 13	33	CH 5 S	48	CH 9 S
4	CH 19 R	19	CNFG 12	34	CH 5 R	49	CH 9 R
5	POWER	20	GROUND	35	POWER	50	POWER
6	CH 20 S	21	CH 0 S	36	CNFG 3	51	CH 10 S
7	CH 20 R	22	CH 0 R	37	CNFG 2	52	CH 10 R
8	CH 21 S	23	CH 1 S	38	CNFG 1	53	CH 11 S
9	CH 21 R	24	CH 1 R	39	CNFG 0	54	CH 11 R
10	POWER	25	POWER	40	GROUND	55	POWER
11	CH 22 S	26	CH 2 S	41	CH 6 S	56	CNFG 7
12	CH 22 R	27	CH 2 R	42	CH 6 R	57	CNFG 6
13	CH 23 S	28	CH 3 S	43	CH 7 S	58	CNFG 5
14	CH 23 R	29	CH 3 R	44	CH 7 R	59	CNFG 4
15	POWER	30	POWER	45	POWER	60	GROUND

Table 2-2. 36-Channel Pin Definitions (P400)							
Pin #	Name	Pin #	Name	Pin#	Name	Pin#	Name
1	CH 24 S	16	CNFG 19	31	CH 34 S	46	CH 14 S
2	CH 24 R	17	CNFG 18	32	CH 34 R	47	CH 14 R
3	CH 25 S	18	CNFG 17	33	CH 35 S	48	CH 15 S
4	CH 25 R	19	CNFG 16	34	CH 35 R	49	CH 15 R
5	POWER	20	GROUND	35	POWER	50	POWER
6	CH 26 S	21	CH 30 S	36	CNFG 23	51	CH 16 S
7	CH 26 R	22	CH 30 R	37	CNFG 22	52	CH 16 R
8	CH 27 S	23	CH 31 S	38	CNFG 21	53	CH 17 S
9	CH 27 R	24	CH 31 R	39	CNFG 20	54	CH 17 R
10	POWER	25	POWER	40	GROUND	55	POWER
11	CH 28 S	26	CH 32 S	41	CH 12 S	56	CNFG 11
12	CH 28 R	27	CH 32 R	42	CH 12 R	57	CNFG 10
13	CH 29 S	28	CH 33 S	43	CH 13 S	58	CNFG 9
14	CH 29 R	29	CH 33 R	44	CH 13 R	59	CNFG 8
15	POWER	30	POWER	45	POWER	60	GROUND

Table 2-3. 72-Channel Pin Definitions (P200)

Pin #	Name	Pin #	Name	Pin#	Name	Pin#	Name
1	CH 36	16	CNFG 15	31	CH 8	46	CH 16
2	CH 37	17	CNFG 14	32	CH 9	47	CH 17
3	CH 38	18	CNFG 13	33	CH 10	48	CH 18
4	CH 39	19	CNFG 12	34	CH 11	49	CH 19
5	POWER	20	GROUND	35	POWER	50	POWER
6	CH 40	21	CH 0	36	CNFG 3	51	CH 20
7	CH 41	22	CH 1	37	CNFG 2	52	CH 21
8	CH 42	23	CH 2	38	CNFG 1	53	CH 22
9	CH 43	24	CH 3	39	CNFG 0	54	CH 23
10	POWER	25	POWER	40	GROUND	55	POWER
11	CH 44	26	CH 4	41	CH 12	56	CNFG 11
12	CH 45	27	CH 5	42	CH 13	57	CNFG 10
13	CH 46	28	CH 6	43	CH 14	58	CNFG 9
14	CH 47	29	CH 7	44	CH 15	59	CNFG 8
15	POWER	30	POWER	45	POWER	60	GROUND

Table 2-4. 72-Channel Pin Definitions (P400)

Pin #	Name	Pin #	Name	Pin#	Name	Pin#	Name
1	CH 48	16	CNFG 19	31	CH 68	46	CH 28
2	CH 49	17	CNFG 18	32	CH 69	47	CH 29
3	CH 50	18	CNFG 17	33	CH 70	48	CH 30
4	CH 51	19	CNFG 16	34	CH 71	49	CH 31
5	POWER	20	GROUND	35	POWER	50	POWER
6	CH 52	21	CH 60	36	CNFG 23	51	CH 32
7	CH 53	22	CH 61	37	CNFG 22	52	CH 33
8	CH 54	23	CH 62	38	CNFG 21	53	CH 34
9	CH 55	24	CH 63	39	CNFG 20	54	CH 35
10	POWER	25	POWER	40	GROUND	55	POWER
11	CH 56	26	CH 64	41	CH 24	56	CNFG 11
12	CH 57	27	CH 65	42	CH 25	57	CNFG 10
13	CH 58	28	CH 66	43	CH 26	58	CNFG 9
14	CH 59	29	CH 67	44	CH 27	59	CNFG 8
15	POWER	30	POWER	45	POWER	60	GROUND

Setting Interrupt Priority Jumpers

At power up, after a SYSRESET, or after resetting the Relay Driver Module via the Status/Control Register, interrupts are enabled. With interrupts enabled, an interrupt is generated approximately 30 msec after any Channel Enable register is accessed (see *Appendix B - Relay Driver Module Register Definitions*). The interrupt priority jumper selects the priority level that will be asserted.

As shipped from the factory, the interrupt priority (IRQ) jumper is set to position 1. For most applications, this priority will not need to be changed. As required, you can set interrupt priorities from 1 to 7 or set level X.

When the IRQ jumpers are set to level X, interrupts are disabled. Interrupts can also be disabled using the Status/Control register. To change the IRO

jumper setting, remove the jumper from its current position and replace it at the desired setting (see Figure 2-2 for jumper location). Consult your mainframe manual to make sure that backplane jumpers are configured correctly.

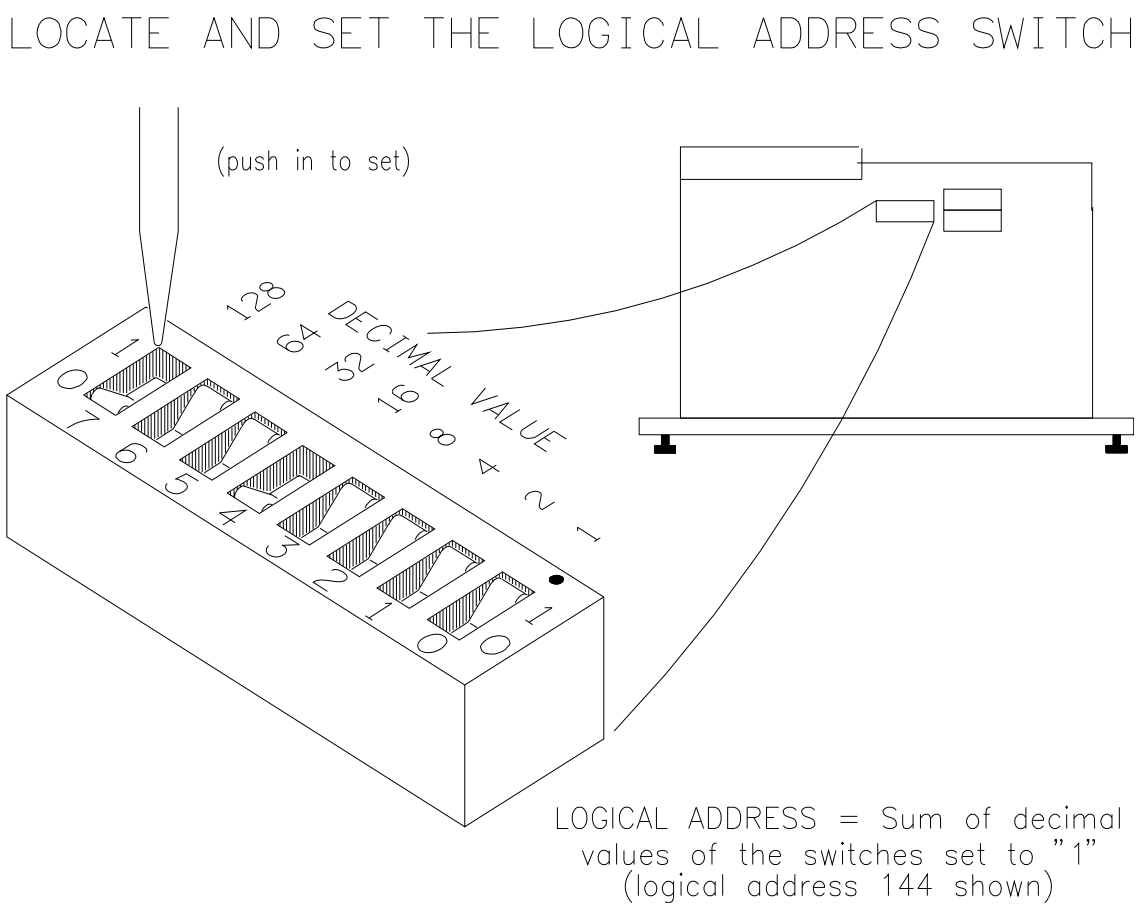


Figure 2-2. Switch/Jumper Locations

Setting Logical Address Switch

Each module within an VXIbus mainframe must be set to a unique logical address. The logical address setting (0 to 255) is controlled by the logical address switch. The factory setting for the Relay Driver is 120. If multiple Relay Driver Modules are to be installed in a mainframe, the logical addresses of some Relay Driver Modules must be changed. See Figure 2-2 for the Logical Address Switch location.

Setting Internal Power Supply Voltage

As shown in Figure 1-1, +5V, +12V, or +24V internal power supplies can be used to drive external loads if required. A maximum of 1 amp is available from the +12V supply. When the Relay Driver Module is used in a VXIbus system with a P2 backplane connector, a maximum of 1 amp is also available from the +5V or +24V supplies. All supplies are fused at 1 amp. The internal power supply voltage value is set with jumpers J200 and J400 (see Figure 2-2 for location). Both jumpers must be set to the same voltage.

Setting Relay Driver Module Operation

This section gives guidelines to select Relay Driver Module operation, including:

- Operations Overview
- Setting 36- or 72-Channel Mode
- Setting Simultaneous/Stepped Actuation (36-Channel Mode)
- Setting Pulsed/Continuous Output (36-Channel Mode)
- Changing Timer Delay

Operations Overview

Table 2-5 summarizes the CNFG bit configurations to set various operating modes. See Tables 2-1 through 2-4 for pin numbers.

Table 2-5. Setting Relay Driver Module Operation

CNFG Bit	P200 Pin #	Connect to GROUND[a]	Leave Open
CNFG 0	39	36-Channel Mode	72-Channel Mode
CNFG 1[b]	38	Stepped Actuation	Simultaneous Actuation
CNFG 2[b]	37	Pulsed Output	Continuous Output
CNFG 3	36	Self-Test Check @ 50 ms	Self-Test Check @ 30 ms
CNFG 4	59	No Self-Test Check	Self-Test Check[c]

a P200 GROUND pin numbers are 20, 40, and 60
b Applicable to 36-channel mode only
c Time (50 msec or 30 msec) depends on CNFG 3 configuration

Setting 36- or 72-Channel Mode

In 72-channel mode, each of the 72 channels operates independently and can be actuated (opened or closed) individually or sequentially. In 36-channel operation, two pins are paired: one for channel SET and one for channel RESET. For example, as shown in Table 2-1, P200 pin #1 is CH 18 S (SET) and pin #2 is CH 18 R (RESET), etc.

Use the CNFG 0 pin (pin 39 on P200) to set 36-channel or 72-channel operation.

- 72-channel Mode: Leave CNFG 0 open
- 36-channel Mode: Connect CNFG 0 to GROUND

Note In some applications, the internal power supply selected may not have enough drive capability to simultaneously activate all 72 (or 36) drivers. In this case, an external power supply should be used so that simultaneous operation can occur.

Setting Simultaneous/ Stepped Actuation (36-Channel Mode)

When 36-channel mode is set (CNFG 0 connected to GROUND), you can use CNFG 1 to set whether to actuate a channel pair at a time with a 30 msec delay between channel pairs (stepped actuation) or to simultaneously actuate all channel pairs in the channel list (simultaneous actuation). When a *RST occurs, up to six relay pairs may be opened at a time.

Use the CNFG 1 pin (pin 38 on P200) to set simultaneous or stepped actuation mode (for 36-channel mode ONLY).

- Simultaneous Actuation: Leave CNFG 1 open
- Stepped actuation: connect CNFG 1 to GROUND

Setting Pulsed/ Continuous Output (36-Channel Mode)

Some microwave relays do not have DC interrupt capability, but the relays are latching. To reduce the power drawn by these relays you may want to use the pulsed mode of operation. With continuous output, currents are continuously applied.

With pulsed output, when a channel is actuated the SET or RESET current is applied until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs. When a *RST occurs, up to six channel pairs at a time may be pulsed. If multiple channel pairs are opened or closed, the next channel pair will wait until the previous channel has finished pulsing.

Use the CNFG 2 pin (pin 37 on P200) to set pulsed or continuous output (for 36-channel mode ONLY).

- Continuous output: Leave CNFG 2 Open
- Pulsed output: Connect CNFG 2 to GROUND

Changing Timer Delay

A 30 msec time delay is supplied for stepped, continuous, or pulsed operation. If your application requires a longer time delay, you could change the components on the board. However, doing this may cause Self Test to fail.

You can use one of two methods to eliminate the Self Test error:

- If the required time delay is 50 msec or less, connecting CNFG 3 to GROUND causes the Self Test to check the timer at 50 msec rather than at 30 msec. (This is the maximum internal time check available with the Command Module.)

- If the required time delay is greater than 50 msec, connecting CNFG 4 to GROUND will skip the Self Test time check completely.

Installing the Relay Driver Module

When you have configured the Relay Driver Module(s) for your application, you can install them in a B-Size or C-Size Mainframe. This section gives guidelines to install one or more Relay Driver Modules, including hardware installation and software installation.

Hardware Installation

Relay Driver Modules can be installed in any slot of a B-Size or C-Size VXIbus or VMEbus mainframe, except slot 0. You can directly install a module in a B-Size mainframe, but you will need an Extender (Agilent E1409A or equivalent) Card installed in a C-Size mainframe for each slot in which a Relay Driver Module will be installed. See the appropriate mainframe *Installation and Getting Started Guide* for procedures to install one or more Relay Driver Modules in the mainframe.

Switchbox Installation

You can install and configure Relay Driver Modules in a switchbox instrument (see "Switchbox Definition" in *Chapter 1 - Getting Started*). Some guidelines for installing Relay Driver Modules in a switchbox instrument follow, depending on whether you want to use Agilent E1339A modules or a combination of Agilent E1339A and Z2309 modules.

Agilent E1339A Switchbox Installation

Two or more Agilent E1339A modules can be combined in a switchbox if the Agilent E1339A SCPI driver is downloaded into the Command Module (see "Downloading the SCPI Driver"). In this case, the Command Module combines all Agilent E1339A cards with sequential logical addresses (such as 120, 121, and 122) into one switchbox instrument.

Note You cannot combine the Agilent E1339A with other types of switch cards, such as the Agilent E1345, to form a single instrument because the Agilent E1339A is not a member of the SWITCHBOX driver.

Agilent E1339/Z2309A Switchbox Installation

If you have existing Agilent Z2309A Relay Driver Modules, you can combine them with Agilent E1339A Relay Driver Modules to form a switchbox, since the hardware for these two cards is identical. The requirements for this type of switchbox are the same as for an Agilent E1339A switchbox.

Software Installation

This section gives guidelines for software installation, including:

- Typical System Configuration
- Installing Agilent E1339A SCPI Driver
- Installing Agilent VXI*plug&play* Drivers
- ZSWITCHBOX Considerations

Typical Software Configuration

To use the Relay Driver Module and to run the example programs in this manual and in the Help system, first download the Agilent E1339A SCPI Driver into the Command Module (Agilent E1406, E1306, or mainframe) and then install the VXI*plug&play* Driver for the Relay Driver Module.

When this is done, you have a choice of two methods of programming the Relay Driver Module. You can program the Relay Driver Module using VXI*plug&play* Function calls, or you can use the downloaded SCPI driver and VISA calls.

Downloading the SCPI Driver

The SCPI driver must be downloaded prior to using VXI*plug&play* drivers to obtain the full functionality of the VXI *plug&play* driver. Failure to download the SCPI driver will result in "Hardware missing" errors on some commands, but you will still be able to open and close channels with the VXI*plug&play* driver.

To download the Agilent E1339A SCPI driver into the Agilent Command Module (Agilent E1406, E1306, or B-Size mainframe) use Agilent VIC for Windows or the driver installer for DOS applications. (These applications are on the CD.)

Installing Agilent VXI*plug&play* Drivers

You can install the Agilent VXI*plug&play* Drivers using the instructions on the Agilent Instrument Drivers CD. It is recommended you create icons for the Relay Driver Soft Front Panel, Driver Help File, Driver Readme Files, and SCPI Example Programs.

You can install the Agilent VXI*plug&play* driver with or without installing the downloaded SCPI driver. However, to obtain the full functionality of the VXI*plug&play* driver, you should first download the Agilent E1339A SCPI driver. See "Downloading the SCPI Driver".

ZSWITCHBOX Driver Considerations

If you do not have any other types of ZSWITCHBOX modules, you should eliminate the ZSWITCHBOX driver so that the Agilent E1339A SCPI driver recognizes the modules, rather than the ZSWITCHBOX driver. This will allow you to use the full capabilities of the Agilent E1339A and Agilent Z2309A modules.

Eliminating ZSWITCHBOX will not affect embedded controller operation, since ZSWITCHBOX was not designed for embedded controllers. However, you can now use the VXI*plug&play* drivers to make function calls to the Agilent E1339A and Z2309 modules.

Verifying Initial Operation

When you have installed and configured the Relay Driver Module for your application, you can use the following example program to verify initial operation of the module.

Example: Reset, Self-Test, and Module ID

This program resets the Relay Driver Module, performs the module self test, and reads the module ID and description.

```
/*  Example: Reset, Self-Test, and Module ID */
#include <visa.h>
#include <stdio.h>
#include <stdlib.h>
#define INSTR_ADDR "GPIB-VXI::120::INSTR" /* 120  is the E1339A*/
                                          /* logical address */

long main()
{
ViStatus errStatus;                       /* status from VISA call */
ViSession viRM;                           /* Resource Mgr. session */
ViSession E1339;                          /* session for E1339A */
char id_string [256] = {0};               /* ID string buffer */
char selftst_string[256] = {0};

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

/* Open the Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

/* Reset the E1339A */
errStatus = viPrintf (E1339, "*RST;*CLS\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Send the Self Test Command */
errStatus = viQueryf (E1339, "*TST?\n","%t",selftst_string);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
   return errStatus;}
printf("Self Test Result is %s\n",selftst_string);

/* Query the ID string */
errStatus = viQueryf (E1339, "*IDN?\n","%t",id_string);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viQueryf() returned 0x%x\n",errStatus);
   return errStatus;}
printf("IDN? returned %s\n",id_string);

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}
}                                         /* End of main program */
```

Chapter 3
Using the Relay Driver Module

Using This Chapter

This chapter provides examples to program and use the Relay Driver Module using Standard Commands for Programmable Instrumentation (SCPI). The chapter contents are:

- Using 72-Channel Mode . . . . . page 32
- Using 36-Channel Mode . . . . . page 33
- Example Programs . . . . . page 37

Note Do not do register writes if you are also controlling the module via the downloaded SCPI driver. Since the SCPI driver will not know the instrument state, an interrupt may occur causing the driver and/or Command Module to fail. See *Appendix B - Relay Driver Module Register Definitions* for information on registers and register-based programming.

Note If you installed Agilent VXI*plug&play* Drivers (see *Chapter 2 - Configuring the Relay Driver Module*), you can use the Soft Front Panels to program and use the Relay Driver Module from a simulated front panel. See "Using the Soft Front Panels" in the VXI*plug&play* Help system for details.

Using 72-Channel Mode

In 72-channel mode, each channel acts independently and channels can be switched (opened/closed) individually or can be scanned. Use OPEN or CLOSe to switch channels or use SCAN to scan channels. Leave CNFG 0 open to set 72-channel mode.

Operations Overview

Table 3-1 summarizes 72-channel operations for switching a single channel, switching several channels, and scanning channels.

Table 3-1. 72-Channel Mode Operations

Switching One Channel	Switching Multiple Channels	Scanning Channels
OPEN (@ccnn) opens the channel and CLOSe (@ccnn) closes the channel.	All channels in OPEN <*channel list*> or CLOSe <*channel_list*> are actuated nearly simultaneously.	A 60-msec delay occurs between actuations of each channel in the SCAN <*channel_list*>.

Switching Channels

For 72-channel mode, **switching channels** means to open one or more open-collector outputs with the OPEN <*channel_list*> command or to close one or more open-collector outputs with the CLOSe <*channel_list*> command. When a channel is closed, current is sunk. When a channel is open, current is not sunk.

When more than one channel is specified in the OPEN or CLOSe command, all channels in the *channel_list* are opened or closed nearly simultaneously (within 100 μsec).

Note When multiple channels are switched, the actuation order is not guaranteed. If actuation order is important, use a separate OPEN or CLOSe command for each channel.

Scanning Channels

For 72-channel mode, **scanning channels** means to open/close more than one channel using the SCAN <*channel_list*> command. When SCAN <*channel_list*> is used, a 60 msec delay occurs between channel actuations. The scan sequence for 72-channel mode when SCAN is used is:

1. Close the first channel in the *channel_list*
2. Wait for the 30 msec timer
3. Open the first channel
4. Wait for the 30 msec timer
5. Repeat steps 1 - 4 for next channel, to the end of the *channel_list*
6. If ARM:COUNt > 1, repeat steps 1 - 5 for ARM:COUNt *number*

If TRIG:SOUR IMM is set, the delay time between channel actuations is 60 msec. If another TRIG *source* is used and the time between triggers is at least 60 msec, the scan advances when the trigger is received. If another TRIG source is used and the time between triggers is <60 msec, a "Trigger Ignored" error is generated.

Using 36-Channel Mode

In 36-channel mode, two bits are paired together and one bit is the inverse of the other. The channel pairs formed are CH 0 SET/CH 0 RESET, CH 1 SET/CH 1 RESET, etc. This mode can be used to drive two coil switches, one for SET and one for RESET.

Note CNFG 0 must be connected to GROUND to set 36-channel mode.

Operations Overview

As shown in Table 3-1, simultaneous actuation, stepped actuation, pulsed output, and continuous output modes are available for 36-channel mode.

Table 3-2. 36-Channel Mode Operations

Operation	Switching One Channel Pair	Switching Multiple Channel Pairs	Scanning Channel Pairs
Simultaneous/Stepped Actuations (Valid only when CNFG 0 is connected to GROUND)			
Simultaneous Actuation (CNFG 1 Open)	OPEN (@ccnn) opens the channel pair and CLOSe (@ccnn) closes the channel pair.	All channel pairs (SET/RESET) in the OPEN/CLOSe *channel_list* are actuated nearly simultaneously.	A 60-msec delay occurs between actuations of each channel pair in the SCAN *channel_list*.
Stepped Actuation (CNFG 1 connected to GROUND)	OPEN (@ccnn) opens the channel pair and CLOSe (@ccnn) closes the channel pair.	A 30-msec delay occurs between actuation of each channel pair in the OPEN/CLOSe *channel_list*.	A 60-msec delay occurs between actuations of each channel pair in the SCAN *channel_list*.
Continuous/Pulsed Outputs (Valid only when CNFG 0 is connected to GROUND)			
Continuous Output (CNFG 2 Open)	The SET or RESET output current is continuously applied to the channel pair when actuated.	The SET or RESET output current is continuously applied to the channel pair that is actuated.	The SET or RESET output current is continuously applied to the channel pair that is actuated.
Pulsed Output (CNFG 2 connected to GROUND)	When the channel pair is actuated, the SET or RESET output current is applied until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs.	The SET or RESET output current is applied to the actuated channel pair until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs. This process is then repeated with the next channel pair in the OPEN/CLOSe *channel_list* until the list is completed.	The SET or RESET output current is applied to the actuated channel pair until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs. For scanning, pulsed output mode requires twice as much time as continuous output to complete the operation.

Setting Simultaneous/ Stepped Actuations

The CNFG 1 bit sets 36-channel mode simultaneous or stepped actuation. For simultaneous actuations, leave CNFG 1 open. To set stepped actuation, connect CNFG 1 to GROUND.

With **simultaneous actuation**, channel pairs are actuated nearly simultaneously (less than 100 μsec) when more than one channel is specified in the *channel_list*.

With **stepped actuation**, an OPEN <*channel_list*>or CLOSe <*channel_list*> command waits for the internal 30 msec timer to interrupt between each channel pair. The instrument will wait for the timer to elapse (causing an IRQ) before the next command is executed.

Note With OPEN and CLOSe, channel actuation order is not guaranteed. If actuation order is important, use a separate OPEN or CLOSe for each channel pair.

Setting Pulsed/ Continuous Outputs

For 36-channel mode, the CNFG2 bit selects pulsed or continous operation. Connect CNFG 2 to GROUND for pulsed output and leave CNFG 2 open for continous output.

With **continuous output**, currents are continuously output when a channel pair specified by OPEN <*channel_list*> or CLOSe <*channel_list*> is actuated.

For **pulsed output**, when a channel pair is opened or closed the SET or RESET current is applied until the 30 msec timer indicates the channel has finished moving. The output current is then removed. On a *RST, up to six relays at a time may be pulsed.

Note Pulsed output requires twice as much time as continuous output to complete the operation.

Switching Channel Pairs

For 36-channel mode, **switching channel pairs** means to open or close one or more channel pairs (SET/RESET). As shown in Table 3-2, when switching more than one channel pair, simultaneous or stepped actuation is available (see "Simultaneous/Stepped Actuations"). Also, continuous or pulsed output mode is available (see "Continuous/Pulsed Outputs").

Use OPEN (@ccnn) to open a single channel pair and use CLOSe (@ccnn) to close a single channel pair. For example, CLOS (@100) closes channel 0 SET and opens channel 0 RESET, while OPEN (@100) opens channel 0 SET and closes channel 0 RESET.

To switch two or more channel pairs, use OPEN <*channel_list*> to open channel pairs and use CLOSe <*channel_list*> to close channel pairs. With **simultaneous actuations**, all channel pairs in the OPEN <*channel_list*> or CLOSe <*channel_list*> are actuated essentially simultaneously (less than μsec). With **stepped actuations**, a 30-msec delay occurs between actuations of each channel pair in the *channel_list*.

Note When multiple channel pairs are switched, the actuation order is not guaranteed. If actuation order is important, use a separate OPEN or CLOSe command for each channel.

Scanning Channel Pairs

For 36-channel mode, **scanning channel pairs** means to open/close more than one channel pair (SET/RESET) using the SCAN <*channel_list*> command to set the scan list. When scanning channel pairs, continuous or pulsed output is available (see "Setting Pulsed/Continuous Outputs"). The scan sequences for continuous and pulsed output follow.

Note The scan sequence depends on whether continuous or pulsed output is selected (as set with CNFG 2), but is not affected by the setting for simultaneous or stepped actuation (as set with CNFG 1).

Scan Sequence: Continuous Output

To set 36-channel mode scanning with continuous output, connect CNFG 0 to GROUND and leave CNFG 2 open. (The CNFG 1 setting does not matter.) With continuous output, a 60 msec delay occurs between actuations of channel pairs for each channel pair in the SCAN <*channel_list*>. The scan sequence for 36-channel mode scanning with continous output is:

1. Close the first channel SET and open the first channel RESET
2. Wait for the 30 msec timer
3. Open the first channel SET and close the first channel RESET
4. Wait for the 30 msec timer
5. Repeat steps 1 - 4 for next channel, to the end of the *channel_list*
6. If ARM:COUNt >1, repeat steps 1 - 5 for ARM:COUNt *number*

If TRIG:SOUR IMM is set, the delay time between channel actuations is 60 msec. If another TRIG *source* is used and the time between triggers is at least 60 msec, the scan advances when the trigger is received. If another TRIG source is used and the time between triggers is <60 msec, a "Trigger Ignored" error is generated.

Scan Sequence: Pulsed Output

To set pulsed output, connect CNFG 0 to GROUND and connect CNFG 2 to GROUND. (The CNFG 1 setting does not matter.) For pulsed output, when a channel pair is opened or closed the SET or RESET current is applied until the 30 msec timer indicates the channel has finished moving. The current is then removed, and another 30 msec wait occurs. On a *RST, up to six channel pairs may be actuated at a time. The scanning sequence for 36-channel mode with pulsed operation is:

1. Close the first channel SET and open the first channel RESET
2. Wait for the 30 msec timer
3. Turn off the first channel SET current output
4. Wait for the 30 msec timer

5. Close the first channel RESET
6. Wait for the 30 msec timer
7. Turn off the first channel RESET current output
8. Wait for the 30 msec timer

9. Repeat steps 1 - 8 for next channel, to the end of the *channel_list*
10. If ARM:COUNt >1, repeat steps 1 - 9 for ARM:COUNt *number*

If TRIG:SOUR IMM is set, the delay time between channel actuations is 120 msec. If another TRIG *source* is used and the time between triggers is at least 120 msec, the scan advances when the trigger is received. If another TRIG source is used and the time between triggers is <120 msec, a "Trigger Ignored" error is generated.

Example Programs

This section provides some example programs for the Relay Driver, including:

- Using the Example Programs
- Example: Closing a Channel (72-Channel Mode)
- Example: Closing a Channel (36-Channel Mode)
- Example: Scanning Channels (72-Channel Mode)
- Example: Scanning Channel (36-Channel Mode)

Using the Example Programs

This chapter, *Chapter 2 - Configuring the Relay Driver Module*, and *Chapter 4 - Understanding the Relay Driver Module* contain some example programs that show ways to use the Relay Driver Module. These examples are written in the C language and require only the downloaded Agilent E1339A SCPI driver or the ZSWITCHBOX SCPI driver.

Copying Example Programs

For the online version of this manual you can copy the example programs to the clipboard and paste them into your application by using the Adobe Acrobat Reader copy and paste functions. To copy an example program from the online version of this manual:

1. Click the Text Selection Tool from the toolbar
2. Highlight the text to be copied
3. Use Edit | Copy to copy the text to the clipboard
4. Insert the clipboard text into your application

Running Example Programs

The example programs in this chapter and in *Chapter 4 - Understanding the Relay Driver Module* were developed with the ANSI C language using the VISA I/O Library. For additonal information, refer to the *Agilent VISA User's Guide.* These programs were written and tested in Microsoft Visual C++, but should compile under any standard ANSI C compiler.

To run these programs, you must have the VISA and an GPIB card installed and properly configured in your PC. The Agilent Command Module (Agilent E1406, E1306, or mainframe) provides direct access to the VXIbus backplane.

Additional Example Programs

See *Chapter 4 - Understanding the Relay Driver Module* for more examples to use the Relay Driver Module. See *Chapter 5 - Relay Driver Module Command Reference* for a description of the SCPI and IEEE-488 Common Commands applicable to the Relay Driver Module. See the online VXI*plug&play* online Help system for VXI*plug&play* programs in Visual C++ and Visual Basic.

Note

The example programs in Chapter *5 - Relay Driver Module Command Reference* include only the SCPI commands for the application. However, you can adapt the example programs in this chapter to the commands listed.

Example: Closing a Channel (72-Channel Mode)

This example shows how to close a channel for the Relay Driver Module in 72-channel mode. For 72-channel mode, OPEN opens the specified channel(s) and CLOSe closes the specified channel(s). If more than one channel is specified, all channels are opened or closed nearly simultaneously, but the actuation order is not guaranteed. To set 72-channel mode, CNFG 0 must not be connected to GROUND.

Typical Connections

Figure 3-1 shows typical connections to Channel 36. The +12V internal supply is applied to the external relay.

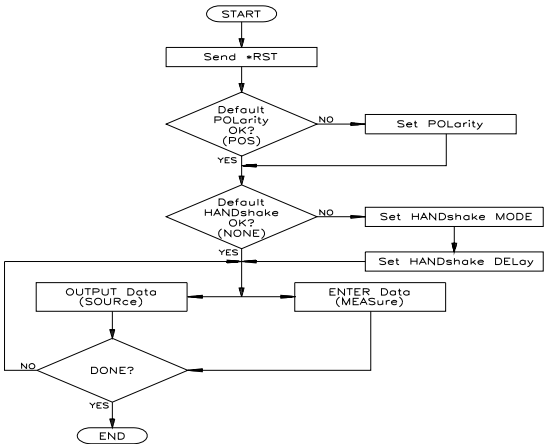


Figure 3-1. Example: Closing a Channel (72-Channel Mode)

SCPI Commands

To close channel 36, the command is CLOSe (@136) where 1 is the card number and 36 is the channel number. When this command is executed, channel 36 is closed, allowing current to be sunk.

Example Program

```
/* Example: Closing a Channel (72-Channel Mode) */
/*This program closes a Relay Driver Module channel in 72-channel mode */

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

#define INSTR_ADDR "GPIB-VXI::120::INSTR"   /* 120 is the E1339A*/
                                            /*logical addr */

long main()
{
ViStatus errStatus;                         /* status from VISA call */
ViSession viRM;                             /* Resource Mgr. session */
ViSession E1339;                            /* session for E1339A */
int ch = 136;                               /* channel = 136 */

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

/* Open the Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Channel 36 */
errStatus = viPrintf (E1339, "CLOS (@%d)\n",ch);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

}                                           /* End of main program */
```

Example: Closing a Channel (36-Channel Mode)

This program shows how to close a channel pair for the Relay Driver Module in 36-channel mode. For 36-channel mode, OPEN opens the specified channel(s) SET and CLOS closes the specified channel(s) SET. If more than one channel pair is specified, all channels are opened or closed nearly simultaneously, but the actuation order is not guaranteed. To set 36-channel mode, CNFG 0 must be connected to GROUND.

Typical Connections

Figure 3-2 shows a Relay Driver Module configured for 36-channel mode (CNFG 0 connected to ground) with typical connections to Channel 0 SET and Channel 0 RESET. For this example, we will assume P2 capability is available so the +5V internal supply is applied to the external relay..

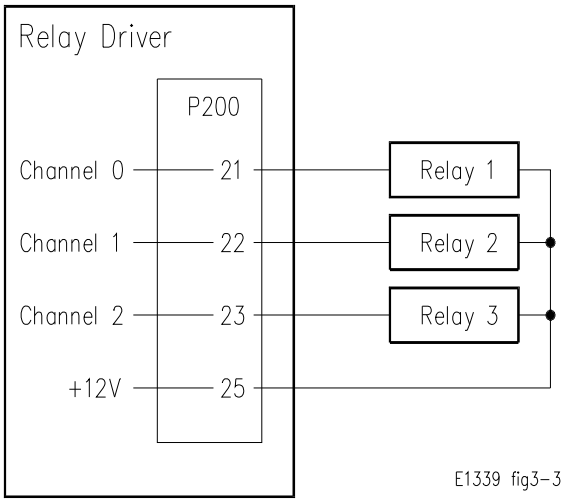


Figure 3-2. Example: Closing a Channel (36-Channel Mode)

SCPI Commands

To close channel pair 0, the command is CLOSe (@100) where 1 is the card number and 00 is the channel pair number. When this command is executed, CH0 SET is closed, allowing current to be sunk and CH0 RESET is opened, not allowing current to be sunk.

Example Program

```
/* Example: Closing a Channel (36-Channel Mode) */
/*This program closes a Relay Driver Module channel in 36-channel mode */

#include <visa.h>
#include <stdio.h>
#include <stdlib.h>
#define INSTR_ADDR "GPIB-VXI::120::INSTR"   /* 120  is the E1339A*/
                                            /* logical addr */

long main()
{
ViStatus errStatus;                         /* status from VISA call */
ViSession viRM;                             /* Resource Mgr. session */
ViSession E1339;                            /* session for E1339A */
int ch = 100;                               /* channel = 100 */

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

/* Open the Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Channel 0 SET, Open Channel 0 RESET */
errStatus = viPrintf (E1339, "CLOS (@%d)\n",ch);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

}                                           /* End of main program */
```

Example: Scanning Channels (72-Channel Mode)

This program shows one way to sequentially actuate three relays by scanning three channels in 72-channel mode.

Typical Connections

Figure 3-3 shows typical connections for channels 0, 1, and 2 to three external relays. We will use the internal +12V supply for external relay power. When scanning, there is an approximate 30 msec delay between channel actuations.

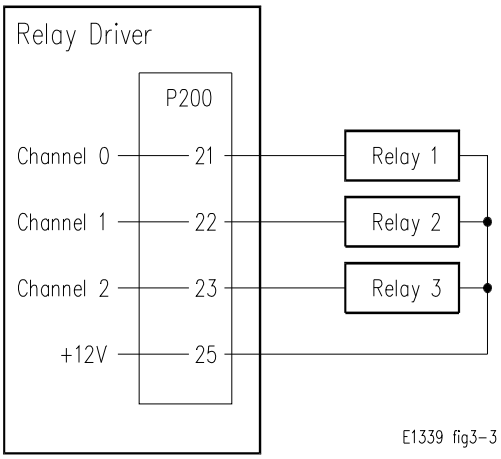


Figure 3-3. Example: Scanning Channels (72-Channel Mode)

SCPI Commands

To scan channels 0 through 2, use SCAN (@100:102) to set the scan list followed by INIT to start the scan. For 72-channel mode, when this command is executed, channel 0 is closed allowing current to be sunk. The scan then advances, opening channel 0 and closing channel 1 and then advances to channel 2. The next advance of the scan opens the last channel. The scan halts after one scan.

Note You can set more than one scan (2 to 32767) or you can set continuous scanning. See *Chapter 4 - Understanding the Relay Driver Module* for details.

Example Program

```
/* Example: Scanning Channels (72-Channel Mode) */
/*This program scans three channels in 72-channel mode */

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

#define INSTR_ADDR "GPIB-VXI::120::INSTR" /* Agilent E1339A  */
                                          /* logical addr */

long main()
{
ViStatus errStatus;                       /* status from VISA call */
ViSession viRM;                           /* Resource Mgr. session */
ViSession E1339;                          /* session for E1339A */
int ch_first = 100;                       /* first scan channel = 100 */
int ch_last = 102;                        /* last scan channel = 102 */

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

/* Open the Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Channels 0 -2 SET, Open Channels 0 - 2 RESET */
errStatus = viPrintf (E1339, "SCAN (@%d:%d);INIT\n",ch_first,ch_last);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

}                                         /* End of main program */
```

Example: Scanning Channels (36-Channel Mode)

This program shows one way to scan three channel pairs in 36-channel mode. For 36-channel mode when a channel pair is actuated, the channel pair SET is closed and the channel pair RESET is opened. After a delay, the channel pair SET is opened and the channel pair RESET is closed. For 36-channel mode scanning, you can set pulsed or continuous output.

Typical Connections

Figure 3-4 shows typical connections to three external relays. We will assume P2 capability is available so the +5V internal supply is applied to the external relays. Connect CNFG 0 and CNFG 2 for the operation desired, as shown in Figure 3-4. For pulsed output, connect CNFG 2 to GROUND. For continuous output, do not connect CNFG 2 to GROUND.

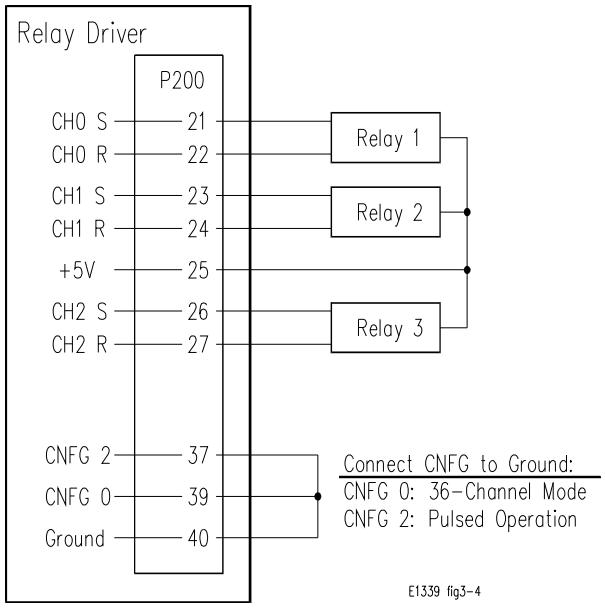


Figure 3-4. Example: Scanning Channels (36-Channel Mode)

SCPI Commands

To scan channel pairs 0 through 2 in 36-channel mode use SCAN (@100:102) to set the scan list, followed by INIT to start the scan. When this command is executed for channel 0, CH0 SET is closed allowing current to be sunk and CH0 RESET is opened, not allowing current to be sunk. The scan then advances to the next channel. The sequence halts after one scan.

Note You can set more than one scan (2 to 32767) or you can set continuous scanning. See *Chapter 4 - Understanding the Relay Driver Module* for details.

Example Program

```
/* Example: Scanning Channels (36-Channel Mode) */
/*This program scans three channels in 36-channel mode */

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

#define INSTR_ADDR "GPIB-VXI::120::INSTR" /*  120  is E1339A */
                                          /*logical addr */

long main()
{
ViStatus errStatus;                       /* status from VISA call */
ViSession viRM;                           /* Resource Mgr. session */
ViSession E1339;                          /* session for E1339A */
int ch_first = 100;                       /* first scan channel = 100 */
int ch_last = 102;                        /* last scan channel = 102 */

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

/* Open the Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Channels 0 -2 SET, Open Channels 0 - 2 RESET */
errStatus = viPrintf (E1339, "SCAN (@%d:%d);INIT\n",ch_first,ch_last);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}
}                                         /* End of main program */
```

Chapter 4
Understanding the Relay Driver Module

Using This Chapter

This chapter gives guidelines to scan Relay Driver Module channels and to use the Scan Complete bit. In addition, this chapter shows how the instrument responds to special IEEE 488.2 commands and to the SYStem:ERRor? command. The chapter contents are:

- Scanning Channels . . . . . page 47
- Setting/Checking States . . . . . page 50
- Example Programs . . . . . page 51

Scanning Channels

This section shows several methods to control scanning channels for the Relay Driver Module, including:

- Setting Scan Mode Operation
- Using Scanning Trigger Sources
- Scanning With External Instruments
- Using the Scan Complete Bit

Setting Scan Mode Operation

For the Relay Driver Module, scanning channels consists of closing and opening a set of channels, one channel at a time. Single-scan, multiple-scan (2 to 32767), or continuous scanning modes are available. See Figure 4-1 for a typical sequence to set Relay Driver Module scanning operation.

Note

Although [ROUTe:]SCAN:MODE NONE|VOLT|RES is accepted by the Relay Driver Module, this command has no effect on Relay Driver Module operation. You can use [ROUTe:]SCAN:MODE? to query the scanning mode (NONE, VOLT, or RES) of the Relay Driver Module.

Using Scanning Trigger Sources

The TRIG:SOUR command specifies the source to advance the scan. You can use the TRIG command to advance the scan when TRIG:SOUR BUS or TRIG:SOUR HOLD is set. Figure 4-2 shows scanning trigger sources.

Sets Number of Scanning Cycles
ARM:COUNt <number> sets 1 to 32767 scanning cycles per INIT command. Default is 1 scanning cycle.

Sets the Trigger Source
TRIG:SOUR defines the trigger source to advance the scan. Sources are:
- BUS = *TRG via GPIB interface
- EXT = Event In connector
- HOLD = Hold Triggering
- IMM = Automatic advance (default)
- TTLT = TTL Trigger Bus Line (0-7)
- ECLT = ECL Trigger Bus LIne (0-1)

Enables Command Module "Trig Out" Port
Enables the Command Module's "Trig Out" port, TTL Trigger bus line (0-7), or ECL Trigger Bus lines (0-1).

Sets Continuous Scanning Cycles
Use INIT ON or INIT 1 to enable continuous cycles. Default is fixed number of scans as determined by the ARM:COUN command.

Selects Scan Mode
Sets NONE|VOLT|RES. This command has no effect on the Relay Driver Module.

Selects the Channel List for Scanning
SCAN <*channel_list*> defines channels to be scanned using the trigger source set by the TRIG:SOUR command.

Enables Scanning
Enables scanning for any trigger source and closes the first channel in the channel list.

Advances Channel List
Advances channel list when trigger source is TRIG:SOUR BUS or TRIG:SOUR HOLD.

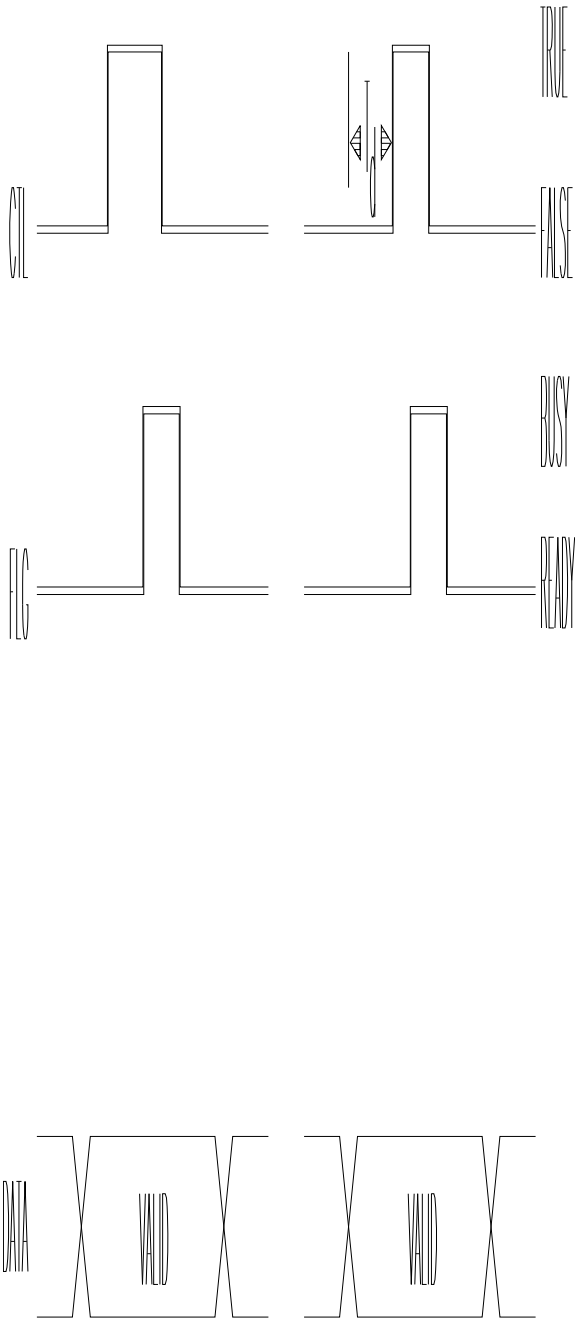


Figure 4-1. Setting Scan Mode Operation

Immediate Triggering (TRIG:SOUR IMM)

TRIG:SOUR IMM sets immediate (internal) triggering. The scan is automatically advanced through the scan list. This is the default trigger mode.

Trigger Hold (TRIG:SOUR HOLD)

TRIG:SOUR HOLD prevents execution of triggers until the trigger source is changed. You can use the TRIG command to trigger a switchbox set to TRIG:SOUR HOLD.

Advancing Scan (TRIG)

You can use the TRIG command to advance the scan list when the switchbox is set for TRIG:SOUR HOLD or TRIG:SOUR BUS. For either trigger source, the scan list advances one channel per TRIG command.

Bus Triggering (TRIG:SOUR BUS)

TRIG:SOUR BUS defines the trigger source as a *TRG or GET command executed over GPIB. With TRIG:SOUR BUS, the scan list is advanced one channel for each *TRG or GET command received.

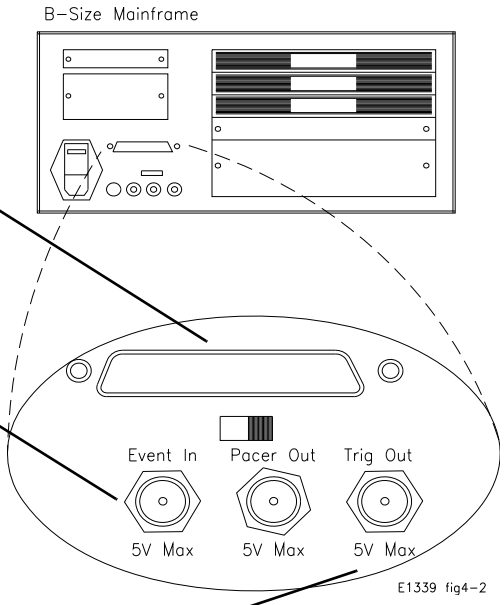
External Triggering (TRIG)

TRIG:SOUR EXT sets external triggering. The trigger source is a (user-supplied) input to the Event In BNC. Use a +5V negative-going pulse to trigger.

With two or more switchboxes in a mainframe, the first switchbox set for EXT trigger keeps the trigger resource until the switchbox source is changed to BUS, HOLD, or IMM.

Enabling Trig Out Port (OUTP ON)

The Trig Out port is shared by all instruments in the mainframe. With the port enabled (with OUTP ON), the port generates a +5V negative-going output trigger pulse each time after a channel closure for ANY switchbox in the mainframe.



NOTE

Operation of switchboxes with other Agilent command modules (e.g., E1406) is the same.

Figure 4-2. Scanning Trigger Sources

Scanning With External Instruments

Scanning Relay Driver Module channels has the same effect as executing multiple OPEN and CLOSe commands. Thus, scanning is useful when a number of relays (or other user devices) are to be closed and opened in sequence. See "Example Programs" in this chapter for an example of scanning using an external instrument.

Using the Scan Complete Bit

You can use the Scan Complete bit (bit 8) in the Operation Status Register of a switchbox to determine when a scanning cycle completes (no other bits in the register apply to the switchbox). See Figure 5-1 in *Chapter 5 - Relay Driver Module Command Reference* for the Status System Register diagrams for the Relay Driver Module. Bit 8 has a decimal value of 256, and you can read it directly with the STAT:OPER? command. See "Example Programs" in this chapter for an example using the Scan Complete bit.

When enabled by the STAT:OPER:ENAB 256 command, the Scan Complete bit will be reported as bit 7 of the Status Byte Register. Use the GPIB Serial Poll or *STB? to read the Status Byte Register.

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

Setting/Checking States

This section describes the response of the Relay Driver Module to three IEEE 488.2 common commands: *SAV, *RCL, and *RST and the response to the error query (SYST:ERR?).

Storing States (*SAV)

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 *SAV are:

- ARM:COUNt
- TRIGger.SOURce
- OUTput:STATe
- INITiate:CONTinuous
- SCAN (the scan list is set to invalid so the command does not save a scan list)
- SCAN:MODE
- SCAN:PORT
- CLOSed Channels

Recalling States (*RCL)

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

Response to Reset (*RST)

The *RST command opens all channels (opens all SET channels in 36-channel mode), invalidates the current channel list for scanning, and sets the following:

- ARM:COUN 1
- TRIG:SOUR IMM
- INIT:CONT OFF
- OUTP:STAT OFF
- SCAN:MODE NONE (no effect on Relay Driver Module operation)
- SCAN PORT NONE (no effect on Relay Driver Module operation)

Error Query Response (SYST:ERR?)

The SYSTem:ERRor? query requests a value from the instrument's error queue. The response takes the following form:

< *err_number* >, < *err_message* >

The < *err_number* > is an integer in the range [-32768 to 32767]. The < *err_message* > is a short description of the error, followed by further information about the error.

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

Example Programs

This section lists two example programs for the Relay Driver Module, including the following. See *Chapter 3 - Using the Relay Driver Module* for information on using these programs.

- Example: Scanning Using Trig Out/Event In Ports
- Example: Scan Complete Interrupt

Example: Scanning Using Trig Out /Event In Ports

This example uses the B-Size mainframe Trig Out and Event In ports to synchronize Relay Driver Module channel closures with an external measurement device (such as an Agilent 34401 voltmeter). 72-channel mode operation is assumed, but you can use the program for 36-channel operation also.

Typical Connections

See Figure 4-3 for typical user connections to a B-Size mainframe.

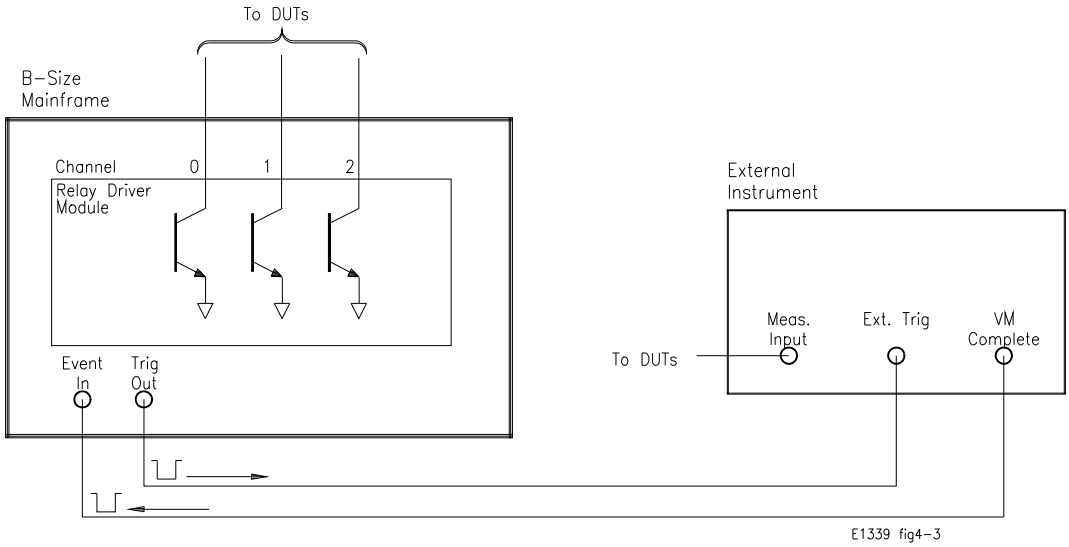


Figure 4-3. Example: Scanning Using Trig Out/Event In Ports

SCPI Commands

For this example, since synchronization with the computer cannot be ensured, the external instrument must have internal memory capacity to store the readings. The sequence of operation and associated SCPI commands are:

1. INIT closes channel 0
2. Channel closure causes trigger to be output from "Trig Out" port
3. Trigger to "Ext Trig In" initiates channel 0 measurement
4. Channel 0 measurement result is stored in external instrument
5. Trigger is then output from "VM Complete" port
6. Trigger to "Event In" port advances scan to channel 1
7. Steps 2-6 are repeated for channels 1 and 2

```
external instrument commands          !Configure ext instrument
*RST;*CLS                             !Reset relay driver module
                                       to known state
OUTP ON                               !Enable Cmd Mod Trig Out port
TRIG:SOUR EXT                         !Event In triggering
SCAN (@l00:102)                       !Scan channels 0 - 2
INIT                                  !Enable scan

Start loop (3 channels)
     Enter results
End Loop
```

Example Program

```
/* Example: Scanning Using Trig Out and Event In Ports */

/*This example uses the Trig Out and Event In ports to synchronize Relay */
/*Driver switch channel closures to an external measurement device */

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

#define INSTR_ADDR1 "GPIB-VXI::120::INSTR" /*120 is * E1339A */
                                            /* logical addr */
#define INSTR_ADDR2 "GPIB-VXI::22::INSTR"  /* 22 is Ext Instrument */
                                            /* logical address*/

long main()
{
ViStatus errStatus;                 /* status from VISA call */
ViSession viRM;                     /* Resource Mgr. session */
ViSession E1339;                    /* session for E1339A */
ViSession INST;                     /*session for Ext Instr*/
int ch_first = 100;                 /* first scan channel = 100 */
int ch_last = 102;                  /* last scan channel = 102 */
int loop;

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

/* Open the External Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR2,VI_NULL,VI_NULL, &INST);
   if (VI_SUCCESS > errStatus){
printf("ERROR: viOpen(%s) returned  0x%x\n",INSTR_ADDR2,errStatus);
   return errStatus;}

/* Configure the External Instrument */
/*External Instrument configuration commands go here*/

/* Open the Relay Driver Module Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR1,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
printf("ERROR: viOpen(%s) returned 0x%x\n",INSTR_ADDR1,errStatus);
   return errStatus;}

/* Reset the Relay Driver Module */
errStatus = viPrintf (E1339, "*RST;*CLS\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Enable the Agilent E1306A Trig Out Port */
errStatus = viPrintf (E1339, "OUTP ON\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
```

```
/* Set Relay Driver Module Trigger Source to External Triggering */
errStatus = viPrintf (E1339, "TRIG:SOUR EXT\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);

/* Set up Scan List*/
errStatus = viPrintf (E1339, "SCAN (@%ld:%ld)\n",ch_first,ch_last);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Enable Scan and Send Trigger to External Instrument*/
errStatus = viPrintf (E1339, "INIT\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Loop to take and enter readings*/

   for(loop=0;loop<(ch_last-ch_first);loop ++)
      {
/* Trigger the external instrument to take a reading */
/* and then take the reading */

   /** External instrument code goes here **/

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (INST);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}
```

Example: Scan Complete Interrupt

This example monitors bit 7 in the Status Register to determine when the cycle completes. The SCPI commands are listed, and an example program follows the SCPI commands.

SCPI Commands

*RST;*CLS	*!Clear all status structure*
STAT:OPER:ENAB 256	*!Enable Scan Complete Bit to set bit 7 in Status Byte Register*
*SRE 128	*!Enable bit 7 of Status Byte Register to assert SRQ*
TRIG:SOUR EXT	*!Set to external trigger mode*
SCAN (@l00:102)	*!Select channels to be scanned*
INIT	*!Start scanning cycle*
Wait for scan complete	*!Enter program lines for computer to do other operations*
Go to SRQ (service request) routine	*!Program goes to this line after interrupt is generated by a completed scanning cycle*

Example Program

```
/* Example: Using Scan Complete Interrupt */
/*This example monitors bit 7 in the Status Byte Register */
/*to determine when the cycle completes. */

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

#define INSTR_ADDR "GPIB-VXI::120::INSTR" /* Agilent E1339A  */
                                          /*logical addr */

long main()
{
ViStatus errStatus;                 /* status from VISA call */
ViSession viRM;                     /* Resource Mgr. session */
ViSession E1339;                    /* session for E1339A */
int ch_first = 100;                 /* first scan channel = 100 */
int ch_last = 102;                  /* last scan channel = 102 */

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

```
/* Open the Instrument Session */
errStatus = viOpen (viRM, INSTR_ADDR,VI_NULL,VI_NULL, &E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viOpen() returned 0x%x\n",errStatus);
   return errStatus;}

/* Reset the Relay Driver Module */
errStatus = viPrintf (E1339, "*RST;*CLS\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Enable Scan Complete Bit to Set Bit 7 in Status Byte Register */
errStatus = viPrintf (E1339, "STAT:OPER:ENAB 256\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Set External Triggering Mode*/
errStatus = viPrintf (E1339, "TRIG:SOUR EXT\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Set up Scan List*/
errStatus = viPrintf (E1339, "SCAN (@%d:%d)\n",ch_first,ch_last);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Enable Scan */
errStatus = viPrintf (E1339, "INIT\n");
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viPrintf() returned 0x%x\n",errStatus);
   return errStatus;}

/* Wait for Scan Complete*/
/*Program lines for computer to do other operations*/

/* Go to Interrupt Routine*/
/*Program goes to this line after scan cycle*/
/* is complete, thus generating an interrupt.*/

/* Close Sessions */
errStatus = viClose (E1339);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}

errStatus = viClose (viRM);
   if (VI_SUCCESS > errStatus){
   printf("ERROR: viClose() returned 0x%x\n",errStatus);
   return 0;}
}
```

Chapter 5
Relay Driver Module Command Reference

Using This Chapter

This chapter describes SCPI (Standard Commands for Programmable Instruments) commands and summarizes IEEE 488.2 Common (*) commands that are applicable to the Relay Driver Module. The chapter includes:

- Command Types. . . . . page 57
- SCPI Command Reference . . . . . page 59
- SCPI Command Quick Reference . . . . . page 90
- IEEE 488.2 Common Commands . . . . . page 91

Note The SCPI commands described in this chapter are available in the downloadable SCPI driver for the Agilent Command Modules such as the Agilent E1306, E1406, and E13XX B-Size Mainframes and the equivalent VXI*plug&play* functions.

See the *Agilent 75000 Series B Mainframe User's Manual* or the *Agilent 75000 Series C Agilent E1406A/B User's Manual* for additional information on SCPI and IEEE 488.2 common commands.

Command Types

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

Common Command Format

The IEEE 488-2 standard defines the Common commands that perform functions like reset, self-test status byte query, etc. Common commands are four or five characters in length, always begin with the asterisk character (*), and may include one or more parameters. The command keyword is separated from the full parameter by a space character. Some examples are *RST; *ESR 32;*STB?

SCPI Command Format

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

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

ROUTe: is the root command, CLOSe and SCAN are second level commands, and :MODE? is a third level command.

Command Separator

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

```
ROUTe:SCAN:MODE?
```

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

Abbreviated Commands

The command syntax shows most commands as a mixture of upper and 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 visibility, you may send the entire command. The instrument will accept either the abbreviated form or the entire command.

For example, if the command syntax shows MEASure, then MEAS and MEASURE are both acceptable forms. Other forms of MEASure, such as MEASU or MEASUR will generate an error. You may use upper or lower case letters. Therefore, MEASURE, measure, and MeAsUrE are all acceptable.

Implied Commands

Implied commands are those which appear in square brackets ([ ]) in the command syntax. (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.

For example, in the partial ROUTe subsystem shown

```
[ROUTe:]
    CLOSe <channel list>
```

the root command ROUTe: is an implied command, so you can use either ROUT:CLOS or CLOS to close the channels specified in the <*channel_list*>.

Parameters

Parameter Types. The following table contains explanations and examples of parameter types you might see in this chapter. Parameters shown within square brackets ([ ]) are optional parameters. (The brackets are not part of the command and are not sent to the instrument.) If you do not specify a value for an optional parameter, the instrument chooses a default value.

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

.

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

Linking Commands

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

*RST;OUTP ON *or* TRIG:SOUR HOLD;*RST

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

ARM:COUN 1;:TRIG:SOUR EXT

SCPI Command Reference

This section describes the Standard Commands for Programmable Instruments (SCPI) commands for the Agilent E1339A/Z2309A Relay Driver Module. Commands are listed alphabetically by subsystem and also within each subsystem.

There are two ways to send commands to the instrument. The most often used way is from a controller over the GPIB interface. This method is referred to as the "GPIB interface" in the command reference. The second way to send commands is from a terminal connected to the Agilent E1406 command module (RS-232). Commands sent this way will be referred to as "from the terminal" in the command reference.

The ABORT command stops a scan in progress when the scan is enabled via the interface and the trigger source is TRIGger:SOURce BUS or TRIGger:SOURce HOLD. See the following comments for ways to stop a scan when the trigger source is not BUS or HOLD.

Syntax ABORt

Comments

- **Channel Status After an ABORT:** Aborting a scan will leave the last channel closed in the closed position.
- Effect on Scan Complete Status Bit: Aborting a scan will not set the "scan complete" status bit.
- **Stopping Scan Enabled from GPIB Interface:** When a scan is enabled from the GPIB interface and the trigger source in not HOLD or BUS, you can clear the interface to stop the command. This can be done by sending an interface CLEAR command (CLEAR 7).

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

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

- **Stopping Scan Enabled From Front Panel**: When a scan is enabled from the front panel or a B-Size mainframe, you can use *RST entered via the interface or the front panel "Reset Instr" or "Clear Instr" keys to stop the scan.
- **Stopping Scans by Using the Terminal:** You may use a terminal connected to the Agilent E1406 command module to stop any scan.

  If the scan was started from the terminal and the trigger source is BUS or HOLD, sent the ABORt command to halt the scan. If the scan was started from the terminal and some other trigger source is being used, a **Ctrl-c** will send an interface CLEAR to the instrument and abort the scan. Sending **Ctrl-r** also sends an interface CLEAR to the instrument and additionally performs a reset (*RST) on the instrument. (See your *Agilent E1406 Command Reference* for details on the terminal interface.)

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

Note Clearing the interface using a **Ctrl-c** from the terminal during a scan leaves the last channel the scan closed in the closed position, and does not set the "scan complete" status bit.

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

Example Program

Example: Stopping a Scan With ABORt

This example stops a (continuous) scan in progress.

```
TRIG:SOUR BUS                    !Bus is trigger source
INIT:CONT ON                     !Set continuous scanning
SCAN (@100:103)                  !Set channel list
INIT                             !Start scanning cycle
  .
  .
ABOR                             !Abort scan in progress
```

The ARM subsystem allows a scan list to be scanned multiple times (1 through 32,767) with one INITiate command.

Syntax

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

ARM:COUNt < *number* > MIN | MAX

Allows scanning cycles to occur a multiple of times (1 to 32767) with one INITiate command when INITiate:CONTinuous OFF |0 is set. MIN sets 1 cycle and MAX sets 32767 cycles.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
number	numeric	1-32767|MIN|MAX	1

Comments

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

Example

Example: Setting Ten Scanning Cycles

This example sets a Relay Driver Module for 10 scans of channels 0 through 3. For 72-channel mode, when the scan sequence completes channels 0 through 3 are closed. For 36-channel mode, when the scan completes channels 0 through 3 SET are closed and channels 0 through 3 RESET are open.

```
ARM:COUN 10                 !Set 10 scans per INIT command
SCAN (@100:103)             !Scan channels 0-3
INIT                        !Start scan, close channel 0
```

ARM:COUNt? [MIN | MAX]

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

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
MIN|MAX	numeric	MIN=1, MAX=32767	current cycles

Comments

Related Commands: INITiate[:IMMediate]

Example

Example: Query Number of Scanning Cycles

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

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

The DISPlay subsystem monitors the channel state of a selected module (or card) in a switchbox. The DISPlay command subsystem only operates with an RS-232 terminal connected to the Agilent E1406 command module's RS-232 port.

Syntax

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

DISPlay:MONitor:CARD <*number*>| AUTO

Selects the module in a switchbox to be monitored. You must use DISP:MON:STAT ON to actually display the monitored module state to the RS-232 terminal.

Parameters

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

Comments

- **Selecting a Specific Module to be Monitored:** Send the card number in a switchbox with the DISPlay:MONitor:CARD command.
- **Selecting the Present Module to be Monitored:** Use the DISPlay:MONitor:CARD AUTO command to select the last module addressed by a switching command (e.g., [ROUTe:]CLOSe).
- ***RST Condition:** DISPlay:MONitor.CARD AUTO

Example

Example: Select Switchbox Module for Monitoring

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

DISPlay:MONitor:CARD?

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

DISPlay:MONitor[:STATe] <*mode* >

Turns the monitor mode ON or OFF. When monitor mode is ON, the RS-232 terminal display presents an array of values indicating the open/close state of every channel on the module. This display is dynamically updated each time a channel is opened or closed.

Parameters

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

Comments

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

- **Typing a Command on a Terminal:** Typing another command on the terminal will cause DISPlay:MONitor[:STATe] to be automatically set to OFF (0). Note that use of the OFF parameter is useful only if the command is issued across the GPIB interface.

- **Monitor Mode on an Agilent E1406 Command Module Display:** A typical display for the Relay Driver Module follows, where all channels are closed. The #H indicates that the data is in hex format. Each channel is represented as a bit in the hex value. The 72 channels are blocked into 4 groups of 16 channels and one group of 8 channels. Closing only channel 3 would appear as 15-0: #H0008. When in the 36-channel mode only the SET bits are displayed. The RESET bits will be the inverse.

  15-0: #HFFFF 31-16: #HFFFF 47-32: #HFFFF 63-48: #HFFFF 71-64: #HFF

  15-0: #HFFFF 31-16: #HFFFF 35-32: #HF

- ***RST Condition:** DISPlay:MONitor[:STATe] OFF | 0. For 72-channel mode, a *RST also opens all channels. For 36-channel mode, a *RST opens all SET channels and closes all RESET channels.

Example

Example: Enabling the Monitor Mode

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

DISPlay:MONitor[:STATe]?

Queries the monitor mode. The command returns "1" if monitor mode is ON or returns "0" if monitor mode is OFF.

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

Syntax

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

INITiate:CONTinuous <*mode*>

Enables or disables continuous scanning cycles for the switchbox.

Parameters

Parameter Name	Parameter 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 a trigger source selected by the TRIGger:SOURce command advances the scan through the channel list.

  For 72-channel mode, a trigger at the end of the channel list closes the first channel in the channel list and the scan cycle repeats for the number of times set by the ARM:COUNt command.

  For 36-channel mode, a trigger at the end of the channel list closes the first SET channel in the channel list and opens the first RESET channel in the channel list. The scan cycle repeats for the number of times set by the ARM:COUNt command.

- **Non-Continuous Scanning Operation:** Non-continuous scanning is enabled with the INITiate:CONTinuous OFF or INITiate:CONTinuous 0 command. Sending the INITiate:IMMediate command closes the first channel in the channel list. Each trigger from a trigger source selected by the TRIGger:SOURce command advances the scan through the channel list.

  For 72-channel mode, at the end of the scanning cycle the last channel in the channel list is opened. For 36-channel mode, at the end of the scanning cycle the last SET channel is the channel list is opened and the last RESET channel in the channel list is closed.

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

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

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

Example **Example: Enabling Continuous Scanning**

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

```
!INIT:CONT ON                 !Enable continuous scanning
SCAN (@100:103)               !Set channel list
INIT                          !Start scan cycle, close channel 0
                              (72-channel mode). Close channel 0
                              SET and open channel 0 RESET
                              (36-channel mode)
```

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 **Example: Query Continuous Scanning State**

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

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

INITiate[:IMMediate]

Starts the scanning process and closes the first channel in the channel list. Successive triggers from the source selected by the TRIGger:SOURce command advances the scan through the channel list. The scan list must be defined before the INIT command will be executed.

Comments

- **Starting the Scanning Cycle:** For 72-channel mode, the INITiate:IMMediate command starts scanning by closing the first channel in the channel list. For 36-channel mode, INITiate:IMMediate starts scanning by closing the first SET channel in the channel list and opening the first RESET channel in the channel list. Successive triggers from the source specified by the TRIGger:SOURce command advances the scan through the channel list. An invalid channel list definition causes an error (see [ROUTe]:SCAN).
- **Stopping Scanning Cycles:** See ABORt.
- **Related Commands:** ABORt, ARM:COUNt, INITiate:CONTinuous, TRIGger, TRIGger:SOURce
- ***RST Condition:** None

Example Example: Starting a Single Scan

This example enables a single scan of channels 0 through 3 of a single-module switchbox. The trigger source to advance the scan is immediate (internal) triggering set with (default) TRIGger:SOURce IMMediate.

```
SCAN (@100:103)                    !Scan channels 0-3 in card 1
INIT                               !Begin scan. For 72-channel mode,
                                   close channel  0 (use immediate
                                   triggering). For 36-channel mode, close
                                   channel 0 SET and open channel 0
                                   RESET.
```

The OUTPut command subsystem enables one Trigger line of the Agilent E1306 or Agilent E1406 Command Module. It can also disable the active line.

Note Do not attempt to turn off a source you have not turned on.

Syntax

```
OUTPut
    :ECLTn(:ECLT0 or :ECLT1)
        [:STATe] <mode>
        [:STATe]?
    [:EXTernal]
        [:STATe] <mode>
        [:STATe]?
    :TTLTrgn(:TTLTrg0 through :TTLTrg7)
        [:STATe] <mode>
        [:STATe]?
```

OUTPut:ECLT*n*[:STATe] <*mode*>

Enables (ON or 1) or disables (OFF or 0) the ECL Trigger bus pulse on the VXI bus line specified by *n*. There are two ECL Trigger lines on the VXI bus allowing valid values for *n* to be 0 and 1. "*mode*" enables (ON or 1) or disables (OFF or 0) the specified ECL Trigger bus line.

Parameters

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

Comments

- **Enabling ECL Trigger Bus:** When enabled, a pulse is output from the selected ECL Trigger bus line (0 or 1) after each channel (or channel pair for 36-channel mode) in the switchbox is closed during a scan. If disabled, a pulse is not output. The output is a negative-going pulse.
- **ECL Trigger Bus Line Shared by Switchboxes:** Only one switchbox configuration can use the selected ECL 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 (or channel pair for 36-channel mode) is closed. To disable the output for a specific switchbox, send the OUTPut:ECLT*n* OFF or 0 command for that switchbox.

- **One Output Selected at a Time:** Only one output (ECLT0 or 1; TTLTrg0, 1, 2, 3, 4, 5, 6, or 7; 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:ECLT*n*[:STATe]?

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

Example

Example: Enabling ECL Trigger Bus Line 0

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

OUTPut:ECLT*n*[:STATe]?

Queries the state of the ECL Trigger bus line specified by *n*. A "1" is returned if the line is enabled. A "0" is returned if the line is disabled. Valid values for *n* are 0 and 1.

Example

Example: Query ECL Trigger Bus Enable State

This example enables ECL Trigger bus line 0 and queries the enable state. The OUTPut:ECLTn? command returns "1" since the port is enabled.

```
OUTP:ECLT0:STAT 1          !Enable ECL Trigger bus line 0.
OUTP:ECLT0?                !Query bus enable state.
```

OUTPut[:EXTernal][:STATe]

Enables or disables the "Trig Out" BNC port on the Command Module to output a trigger when a channel (or channel pair for 36-channel mode) is closed during a scan.

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

Parameters

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

Comments

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

- **Drivers Must be Downloaded (Does not Apply to Embedded Computers):** OUTPut[:EXTernal][:STATe] is available only if the ZSWITCHBOX or Agilent E1339 driver has been downloaded to the Agilent E1306A or Agilent E1406 Command Module. If you do not use the Command Module or have not downloaded one of these drivers to the Command Module, a "Hardware Missing" error is generated.

- **Enabling "Trig Out" Port:** When enabled, a pulse is output from the "Trig Out" BNC port after each scanned switchbox channel is closed. If disabled, a pulse is not output from the port after channel closures. The output pulse is a negative-going pulse.

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

- **One Output Selected at a Time:** Only one output (ECLT0 or 1; TTLTrg0, 1, 2, 3, 4, 5, 6, or 7; 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[:EXTernal][:STATe]?

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

Example **Example: Enabling "Trig Out" Port**

```
OUTP EXT 1                    !Enable "Trig Out" port to output a pulse
                              after each scanned channel (or channel
                              pair) is closed.
```

OUTPut[:EXTernal][:STATe]?

Queries the present state of the "Trig Out" port on the Agilent E1306 or Agilent E1406 Command Module. The command returns "1" if the port is enabled or "0" if the port is disabled.

Example **Example: Query "Trig Out" Port State**

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

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

OUTPut:TTLTrg*n*[:STATe] <*mode*>

Selects and enables which TTL Trigger bus line (0 to 7) will output a trigger when a channel (or channel pair for 36-channel mode) is closed during a scan. This is also used to disable a selected TTL Trigger bus line. "*n*" specifies the TTL Trigger bus line (0 to 7) and "*mode*" enables (ON or 1) or disables (OFF or 0) the specified TTL Trigger bus line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
n	numeric	0 to 7	N/A
<*mode*>	boolean	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 (or channel pair for 36-channel mode) in the switchbox is closed during a scan. If disabled, a pulse is not output. The output is a negative-going pulse.
- **TTL Trigger Bus Line Shared by Switchboxes:** Only one switchbox configuration can use the selected TTL 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 (or channel pair for 36-channel mode) 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 (ECLT0 or 1; TTLTrg0, 1, 2, 3, 4, 5, 6, or 7; 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**: [:ROUT:]SCAN, TRIGger:SOURce, OUTPut:TTLTRG*n*[:STATe]?
- ***RST Condition:** OUTPut:TTLTrg*n*[:STATe] OFF (disabled)

Example

Example: Enabling TTL Trigger Bus Line 7

```
OUTP:TTLT7:STAT 1          !Enable TTL trigger bus line 7 to
                           output a pulse after each scanned channel
                           (or channel pair) is closed.
```

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 Example: Query TTL Trigger Bus Enable State

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

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

The ROUTe command subsystem controls switching and scanning operations for Relay Driver Modules in a switchbox.

Syntax

[ROUTe:]
CLOSe < *channel_list* >
CLOSe? < *channel_list* >
OPEN <*channel_list* >
OPEN? <*channel_list* >
SCAN <*channel_list* >
:MODE <*mode*>
:MODE?
:PORT <*port*>
:PORT?

Note Although [ROUTe:]SCAN:MODE, [ROUTe:]SCAN:MODE?, [ROUTe:] SCAN:PORT, and ROUTe:]SCAN:PORT are accepted by the Relay Driver Module, these commands have no effect on Relay Driver Module operation and are not described in this command reference.

For all [ROUTe:] commands, the term "**close a channel**" means to close an individual channel in 72-channel mode or to simultaneously close the SET channel and open the RESET channel in 36-channel mode.

For all [ROUTe:] commands, the term "**open a channel**" means to open an individual channel in 72-channel mode or to simultaneously open the SET channel and close the RESET channel in 36-channel mode.

[ROUTe:]CLOSe <*channel_list*>

Closes the Relay Driver Module channels specified by *channel_list*. *Channel_list* has the form (@ccnn), (@ccnn,ccnn), or (@ccnn:ccnn) where cc = card number (01-99) and nn = channel number (00-71 for 72-channel mode or 00-35 for 36-channel mode).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
channel_list	numeric	cc00-cc71 (72-channel mode) cc00-cc35 (36-channel mode)	N/A

Comments

- **Closing Channels:** To close:

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

- **Closure Order not Guaranteed**: Closure order for multiple channels with a single CLOS command is not guaranteed. A list of channels will not all close simultaneously. The order in which channels close when specified from a single command is not guaranteed. Use sequential CLOSe commands if needed.

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

- ***RST Condition:** All channels are open in 72-channel mode. In 36-channel mode, all SET channels are open and all RESET channels are closed.

Example

Example: Closing Relay Driver Module Channels

This example closes channels 00 and 13 of a two-module switchbox (card numbers 01 and 02).

```
CLOS (@100,213)
```

! Close channels 100 and 213. In 72-channel mode, 100 closes channel 0 of card #1 and 213 closes channel 13 of card #2.

! In 36-channel mode,100 closes channel 0 SET and opens channel 0 reset, while 213 closes channel 13 SET and opens channel 13 RESET for card #2.

[ROUTe:]CLOSe? <*channel_list* >

Returns the current state of the channel(s) queried. *Channel_list* has the form (@ccnn) where cc = card number (01-99) and nn = channel number (00-71 for 72-channel mode or 00-35 for 36-channel mode). The command returns "1" if channel(s) are closed or returns "0" if channel(s) are open.

Comments

- **Query is Software Readback:** The [ROUTe:]CLOSe? command returns the current software state of the channel(s) specified. It does not account for hardware failures or for channels opened or closed by direct register access (see *Appendix B - Relay Driver Module Register Definitions*).

Example Example: Query Channel Closure

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

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

[ROUTE:]OPEN <*channel_list*>

Opens the Relay Driver Module channels specified by *channel_list. Channel_list* has the form (@ccnn), (@ccnn,ccnn), or (@ccnn:ccnn) where cc = card number (01-99) and nn = channel number (00-71 for 72-channel mode or 00-35 for 36-channel mode).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
channel_list	numeric	cc00-cc71 (72-channel mode) cc00-cc35 (36-channel mode)	N/A

- **Opening Channels:** To open:

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

- **Opening Order not Guaranteed**: Opening order for multiple channels with a single OPEN command is not guaranteed. A list of channels will not all open simultaneously. The order in which channels open when specified from a single command is not guaranteed. Use sequential OPEN commands if needed.

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

- ***RST Condition:** All channels open for 72-channel mode. In 36-channel mode, all SET channels are opened and all RESET channels are closed.

Example Example: Opening Switchbox Channels

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

```
OPEN (@100,213)          !Open  channels 100 and 213. In
                         72-channel mode,100 opens channel 0
                         of  card  #1 and 213 opens channel 13
                          of card  #2.
                         !In 36-channel mode, 100 opens channel
                         0 SET and closes channel 0 RESET of
                         card #1, while 213 opens channel 13 SET
                         and closes channel 13 RESET of card #2.
```

[ROUTe:]OPEN? < *channel_list* >

Returns the current state of the channel(s) queried. *Channel_list* has the form (@ccnn), (@ccnn,ccnn), or (@ccnn:ccnn) where cc = card number (01-99) and nn = channel number (00-71 for 72-channel mode or 00-35 for 36-channel mode). The command returns "1" if channel(s) are open or returns "0" if channel(s) are closed.

Comments

- **Query is Software Readback:** The [ROUTe:]OPEN? command returns the current software state of the channels specified. It does not account for hardware failures or for channels opened or closed by direct register access (see *Appendix B - Relay Driver Module Register Definitions*).

Example

Example: Query Channel Open State

This example opens channels 100 and 213 of a two-module switchbox and queries channel 213 state. Since channel 213 is programmed to be open, "1" is returned. For 72-channel mode, this indicates channel 13 in card #2 is open. For 36-channel mode, this indicates channel 13 SET in card #2 is open, while channel 13 RESET is closed.

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

[ROUTE:]SCAN < *channel_list* >

Defines the channels to be scanned. *Channel_list* has the form (@ccnn), (@ccnn,ccnn), or (@ccnn:ccnn) where cc = card number (01-99) and nn = channel number (00-71 for 72-channel mode or 00-35 for 36-channel mode).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
channel_list	numeric	cc00-cc71 (72-channel mode) cc00-cc35 (36-channel mode)	N/A

Comments

- **Defining Scan List:** When [ROUTe:]SCAN is executed, the channel list is checked for valid card and channel numbers. An error is generated for an invalid channel list.
- **Scanning Channels:** You can scan single channels (@ccnn); multiple channels (@ccnn,ccnn ....); sequential channels (@ccnn:ccnn); groups of sequential channels (@ccnn:ccnn,ccnn:ccnn); or any combination.
- **Scanning Operation:** When a valid channel list is defined, INITiate[:IMMediate] begins the scan and closes the first channel in the channel list. Successive triggers from the source specified by TRIGger:SOURce advance the scan through the channel list.

- **Stopping Scan:** See ABORt.
- **Related Commands:** [ROUTe:]CLOSe, [ROUTe:]OPEN, TRIGger, TRIGger:SOURce
- ***RST Condition:** All channels open for 72-channel mode. For 36-channel mode, all SET channels are open and all RESET channels are closed

Example Example: Scanning Using External Devices

This example shows one way to scan channels using an Agilent E1406 command module via GPIB and an external instrument. The example uses a pulse from the command module "Trig Out" port to synchronize the external instrument to a Relay Driver Module.

```
external instrument commands        !Commands to configure external
                                     instrument
OUTP ON                             !Enable "Trig Out" port on  command
                                     module
TRIG:SOUR BUS                       !Set switchbox to receive Bus triggers
SCAN (@100:107)                     !Set the channel list for chs 0 through 7
INIT                                !Start the scanning cycle
Loop                                !Start the counting loop
    enter result                    !Enter results into computer
    TRIGGER                         !Trigger the switchbox to advance the
                                     channel scan
    increment count                 !Increment the loop
End Loop                            !End of count loop
END                                 !End of program
```

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.

Syntax

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

Note Although STATus:QUEStionable... is supported to provide SCPI compliance, this command does not affect Relay Driver Module operation, and is not described in this command reference.

The STATus system contains four registers, two of which are under IEEE 488.2 control: the Standard Event Register (*ESE?) and the Status Byte Register (*STB?) (see Figure 5-1). 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). The registers are queried using decimal weighted bit values. The decimal equivalents for bits 0 through 15 are included in Figure 5-1.

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

The decimal values are also used in the inverse manner to determine which bits are set from the total value returned by an EVENt or CONDition query. The Relay Driver Module uses only bit 8 of the Operation Status Register. This bit is called the Scan Complete bit, and is set whenever a scan operation completes.

Since completion of a scan operation is an event in time, bit 8 will never appear set when STAT:OPER:COND? is queried. However, you can find bit 8 set with the STAT:OPER:EVEN? query command.

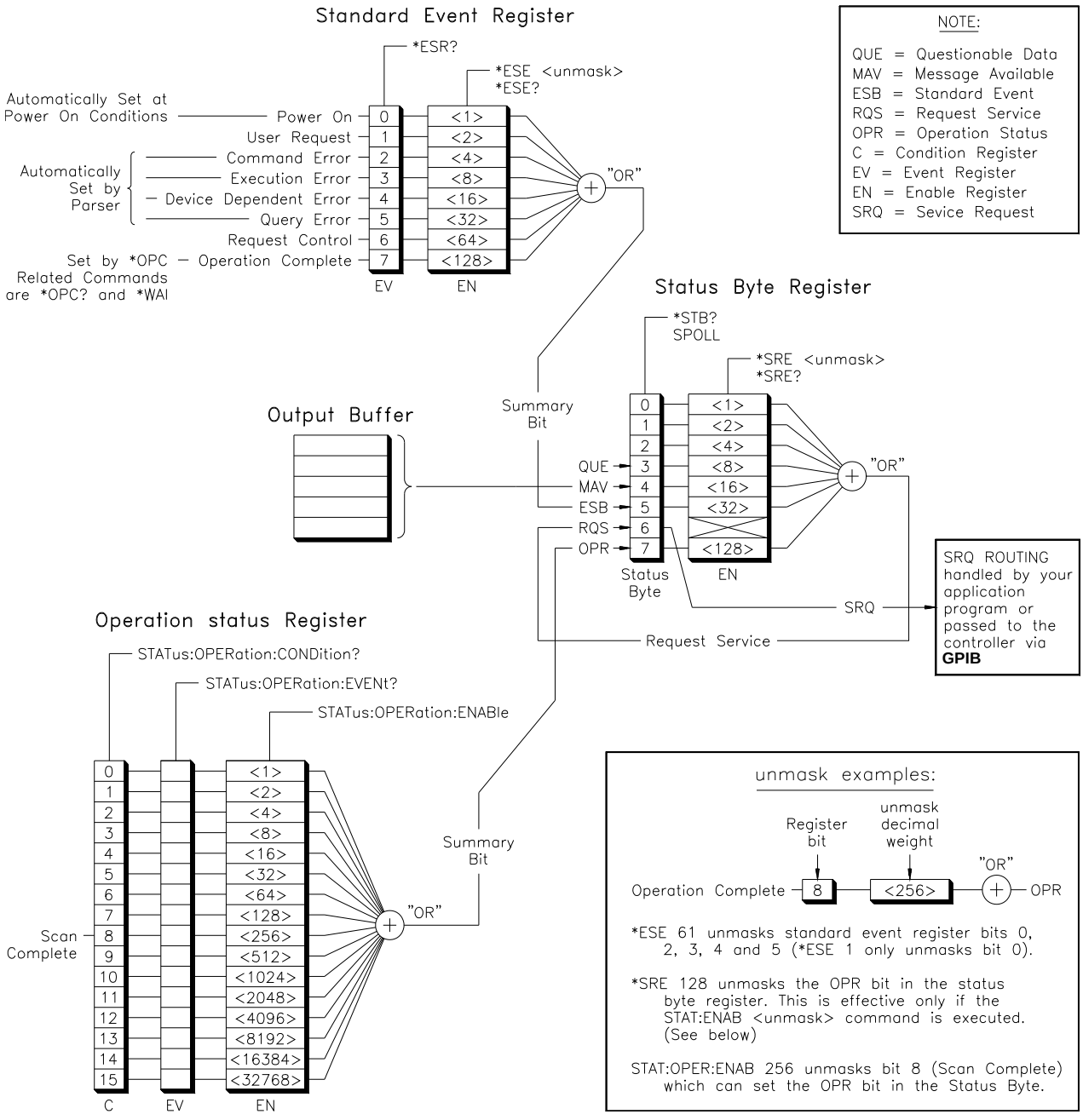


Figure 5-1. Relay Driver Module Status System Register Diagram

STATus:OPERation:CONDition?

Returns the state of the Condition Register in the Operation Status Group. The state represents conditions that are part of the instrument's operation. The Relay Driver Module does not set bit 8 in this register (see STATus:OPERation[:EVENt]?.

STATus:OPERation:ENABle *<unmask>*

Sets an enable mask to allow events recorded in the Event Register to send a summary bit to the Status Byte Register (bit 7). For Relay Driver Modules, when bit 8 in the Operation Status Register is set to 1 and that bit is enabled by the STATus:OPERation:ENABle command, bit 7 in the Status Byte Register is set to 1.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Value
<*unmask*>	numeric	-32768 to 32767	N/A

Comments

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

Example

Example: Enabling Status Byte Register Bit 8

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

STATus:OPERation:ENABle?

Returns which bits in the Event Register (Operation Status Group) are unmasked.

Comments

- **Output Format:** Returns a decimal weighted value from -32768 to 32767 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

Example: Query the Operation Status Enable Register

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

STATus:OPERation[:EVENt]?

Returns which bits in the Event Register (Operation Status Group) are set. The Event Register indicates a time-related instrument event has occurred. Only bit 8 is used by the Relay Driver Module.

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 STATus:OPERation[:EVENt]?**: This 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 the Event Register.
- **Aborting a Scan:** Aborting a scan will leave bit 8 set to 0.
- **Related Commands:** [ROUTe:] SCAN

Example

Example: Reading the Operation Status Register

```
STAT:OPER?                    !Returns the bit values of the Operation
                              Status Register
read the register value       !Returns +256 if bit 8 is set to 1 or
                               returns +0 if bit 8 is set to 0.
```

STATus:PRESet

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

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

Syntax

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

SYSTem:CDEScription? < *number* >

Returns the description of a selected module (card) in a switchbox.

Parameters

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

Comments

- **Relay Driver Module Description:** The SYSTem:CDEScription? < *number* > command returns: `"72 Channel Open Collector Digital Output/Relay Driver"` when in the 72-channel mode. In the 36-channel mode, `"36 Channel uWave Switch Driver"` is returned.
- **36/72 Channel Mode Selection:** The configuration bit is read at power-up, and can be determined by using the SYSTem:CDEScription? < *number* > command.

Example **Example: Reading Card #1 Description**

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

SYSTem:CPON < *number* > | ALL

Sets the selected module (card) or all modules in a switchbox to their power-on state.

Parameters

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

Comments

- **Differences Between *RST and CPON:** *RST and SYSTem:CPON ALL opens all channels of all modules in a switchbox, while SYSTem:CPON < *number* > opens the channels in only the module (card) specified in the command. In addition, CPON does not reset other values such as ARM:COUNt.

Example

Example: Setting Card to its Power-On State

```
SYST:CPON 1                    !Sets module #1 channels to power-on
                               state. For 72-channel mode, all channels
                               are open. For 36-channel mode, all  SET
                               channels are open and all RESET
                               channels are closed.
```

SYSTem:CTYPE? < *number* >

Returns the module (card) type of a selected module in a switchbox.

Parameters

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

Comments

- **Relay Driver Model Number:** The SYSTem:CTYPe? < *number* > command returns:

  `HEWLETT-PACKARD,E1339A/Z2309A,0,`*`revision`*

  where the 0 after E1339A/Z2309A is the module serial number (always 0) and *revision* is the module revision code number.

Note

If you are using the ZSWITCHBOX driver, the string will be:

"HEWLETT-PACKARD,Z2309A,0,*revision"*

It is advisable to upgrade to the Agilent E1339A driver to use all the features of the Agilent E1339A Relay Driver Module described in this manual.

Example

Example: Reading Card #1 Model Number

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

SYSTem:ERRor?

Returns the error numbers and corresponding error messages in the error queue of a switchbox. See *Appendix C - Relay Driver Module Error Messages* for a listing of some switchbox error numbers and messages.

Comments

- **Error Numbers/Messages in the Error Queue**: Each error generated by a switchbox stores an error number and corresponding error message in the error queue. The error message can be up to 255 characters long.
- **Clearing the Error Queue:** An error number/message is removed from the queue each time the SYSTem:ERRor? command is sent. The errors are cleared first-in, first-out. When the queue is empty, each following SYSTem:ERRor? command returns +0, "No error". To clear all error numbers/messages in the queue, execute the *CLS command.
- **Maximum Error Numbers/Messages in the Error Queue:** The queue holds a maximum of 30 error numbers/messages for each switchbox. If the queue overflows, the last error number/message in the queue is replaced by -350, "Too many errors". The least recent error numbers/messages remain in the queue and the most recent are discarded.

Example

Example: Reading the Error Queue

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

The TRIGger command subsystem controls the triggering operation of a Relay Driver Module in a switchbox.

Syntax

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

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 a switchbox with the TRIGger[:IMMediate] command.
- **Related Commands**: INITiate, [ROUTe:]SCAN, TRIGger:SOURce

Example

Example: Advancing Scan Using TRIGger

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

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

TRIGger:SOURce <*source*>

Specifies the trigger *source* to advance the channel list during scanning.

Parameters

Parameter Name	Parameter Type	Parameter Description
BUS	discrete	*TRG or GET command
ECLT*n*	numeric	ECL Trigger bus line 0 or 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 INITtate[:IMMediate] command enables the trigger source. The trigger source must be selected using the TRIGger:SOURce command before executing the INIT command.

- **One Trigger Input Selected at a Time:** Only one input (ECLT0 or1; 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 the TRIGger Command:** You can use TRIGger[:IMMediate] to advance the scan when TRIGger:SOURce BUS or TRIGger:SOURce HOLD is set.

- **Drivers Must be Downloaded (Does not Apply to Embedded Computers):** TRIG:SOURce EXTernal is available only if the ZSWITCHBOX or Agilent E1339 driver has been downloaded to the Agilent E1406 Command Module. If you do not use the Command Module or have not downloaded one of these drivers to the Command Module, a "Trigger source non-existent" error is generated.

- **Using External Trigger Inputs:** With TRIGger:SOURce EXTemal selected, only one switchbox at a time can use the external trigger input at the E1406 Command Module "Trig In" port. The trigger input is assigned to the first switchbox that requested the external trigger source (with a TRIGger:SOURce EXTernal command).

- **Using TTL or ECL Trigger Bus Inputs:** These triggers are from the VXI backplane trigger lines ECL[0,1] and TTL[0-7]. These triggers may be used to trigger the Agilent E1339A driver from other VXI instruments.

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

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

- **Using EXTernal, TTLTrg*n*, and ECLT*n* Trigger Inputs:** After using TRIGger:SOURce EXT | TTLTrg*n* |ECLT*n*, the selected trigger source remains assigned to the Agilent E1339A driver until it is relinquished through use of the TRIG:SOUR BUS|HOLD|IMM command.

  While the trigger is in use by the E1339A driver, no other drivers operating on the Agilent E1306 or Agilent 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 Agilent E1339A driver.

  You should always release custody of trigger sources after completion of an activity by setting the trigger source to BUS, HOLD, or IMMediate (i.e., TRIG:SOUR BUS|HOLD|IMM).

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

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

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

Examples

Example: Scanning Using External Triggers

This example uses external triggering (TRIG:SOUR EXT) to scan channels 0 through 3 of a single-module switchbox. The trigger source to advance the scan is the input to the "Trig In" port on an Agilent E1406 Command Module. When INIT is executed, the scan is started and channel 00 is closed. Then, each trigger received at the "Trig In" BNC port advances the scan to the next channel.

```
TRIG:SOUR EXT                  !Select external triggering
SCAN (@100:103)                !Set channel list
INIT                           !Start scanning cycle
trigger externally             !Advance scan to next channel
```

Example Programs: Scanning Using Bus Triggers

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

```
TRIG:SOUR BUS                 !Select interface (bus) triggering
SCAN (@100:103)               !Set channel list
INIT                          !Start scanning cycle
*TRG                          !Advance scan using bus triggering
```

TRIGger:SOURce?

Returns the current trigger source for the switchbox. This command returns BUS, EXT, HOLD, IMM, TTLT0-7, or ECLT0-1 for sources BUS, EXTernal, HOLD, IMMediate, TTLTrg*n*, or ECLTrg*n*, respectively.

Example

Example: Query Trigger Source

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

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

.
```

Command Subsystem		Description
ABORt		Abort a scan in progress
ARM	:COUNt <*number* > :COUNt? [MIN|MAX]	Multiple scans per INIT command Query number of scans
DISPlay	:MONitor:CARD <*number*>|AUTO :MONitor:CARD? :MONitor[:STATe] <*mode*> :MONitor[:STATe]?	Selects module to be monitored Query the card number Selects monitor mode Query the monitor state
INITiate	:CONTinuous ON|OFF :CONTinuous? [:IMMediate]	Enables/disables continuous scanning Queries continuous scan state Starts a scanning cycle
OUTPut	:ECLT*n*[:STATe] ON|OFF|1|0 :ECLTn[:STATe]? [:EXTernal][:STATe] ON|OFF|1|0 [:EXTernal][:STATe]? :TTLTrg*n*[:STATe] ON|OFF|1|0 :TTLTrgn[:STATe]?	Enables/disables the specified ECL trigger line Queries the specified ECL trigger line Enables/disables command module "Trig Out" port Queries external state Enables/disables the specified TTL trigger line Queries the specified TTL trigger line
[ROUTe:]	CLOSe <*channel_list*> CLOSe? <*channel_list*> OPEN <*channel_list*> OPEN? <*channel_list*> SCAN <*channel_list*>	Closes channel(s) Queries channel(s) closed Opens channel(s) Queries channel(s) opened Defines channels for scanning
STATus	:OPERation:CONDition? :OPERation:ENABle <*unmask*> :OPERation:ENABle? :OPERation[:EVENt]? :PREset	Returns contents of the Operation Condition Register Enables events in 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 to 0
SYSTem	:CDEScription? <*number*> :CPON <*number*>|ALL :CTYPe? <*number*> :ERRor?	Returns description of module in switchbox Sets specified module to its power-on state Returns the module type Returns error number/message
TRIGger	[:IMMediate] :SOURce BUS :SOURce ECLTrg*n* :SOURce EXTernal :SOURce HOLD :SOURce IMMediate :SOURce TTLTrg*n* :SOURce?	Causes a trigger to occur Trigger source is *TRG Trigger source is VXIbus ECL trigger bus line *n* Trigger source is "Event In" port on Agilent E1306/E1406 Holds off triggering Trigger source is the internal triggers Trigger is the VXIbus TTL trigger bus line *n* Queries scan trigger source

IEEE 488.2 Common Commands

The following table describes the IEEE 488.2 Common (*) Commands that apply to the Relay Driver Module. These commands apply to many instruments and many have no specific action related to this module. Only the commands that are related to this module are documented here. For more information on the Common Commands, refer to the *Agilent E1406 Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.

Note If "ZSWITCHBOX" appears in the string returned by *IDN?, you are using the old ZSWITCH driver. It is advisable to upgrade to the Agilent E1339A driver to obtain the full feature set listed in this manual.

Command	Description
*CLS	Clears all status registers and the error queue. See STATus:OPERation[:EVENt]?.
*ESE <*register value*>	Event status enable command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*ESE?	Event status enable query command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*ESR?	Event status register query command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*IDN?	Queries the identity string of the instrument. Returns an arbitrary ASCII response with manufacturer, instrument name, serial number, and revision, similar to that shown. The serial number is returned as 0 since a switchbox can be made up of multiple modules. The revision code will vary as updates are made. (See note above). `HEWLETT-PACKARD,E1339A/Z2309A, 0, revision`
*OPC	Operation complete command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*OPC?	Operation complete command query. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*RCL <*numeric state*>	The *RCL < *numeric_state* > command recalls the state when the last *SAV was executed for the specified numeric state parameter (0-9). If no *SAV was executed for the numeric state, *RST default settings are used. Refer to the *SAV settings list for the settings recalled by *RCL.

Command	Description
*RST	Opens all channels, invalidates the current channel list for scanning, sets the following states, and sets the trigger state to idle. ARM:COUNt 1 TRIGger:SOURce IMM INIT:CONTinuous OFF OUTPut:STATe OFF SCAN MODE:NONE SCAN PORT:NONE
*SAV <*numeric state*>	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 are: ARM:COUNt TRIGger:SOURce OUTput:STATe INITiate:CONTinuous SCAN (the scan list is set to invalid, so the command does not save a scan list) SCAN:MODE SCAN:PORT
*SRE <*register value*>	Service request enable command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*SRE?	Service request enable query command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*STB?	Status byte query command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*TRG	Trigger command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.
*TST?	Performs self test on the instrument and returns a numeric test code. A "0" indicates the test passed, while a non-zero response indicates the Agilent E1339A failed. See to the appropriate User's Manual for these failures. If an Agilent E1339A module fails, a 3- or 4-digit error code is returned in the form [c]cnn, where [c]c indicates card number of failed card (1 - 99) and nn indicates error code +0 - self-test passes +cc01 - firmware error +cc02 - VME bus error while accessing card +cc03 - the card ID read back was in error +cc04 - card registers are bad +cc05 - hardware and firmware have different values. Possibly a hardware error or an outside entity is register-programming the Relay Driver Module. +cc10 - card did not generate an interrupt +cc11 - the busy bit was not held the appropriate length of time, either too short or too long. +cc13 - card generated an interrupt too soon
*WAI	Wait to continue command. See the *Command Module User's Manual* or the ANSI/IEEE Standard 488.2-1987.

Appendix A
Relay Driver Specifications

Item	Specification
Maximum Open Collector Voltage	32V
Sink Current Capability	200 mA per switch
Maximum Internal Power Supply Current	+5V @ 1A (fused and requires P2) +12V @ 1A (fused) +24V @ 1A (fused and requires P2) One of the three internal power supplies can be selected. Use of the internal power supply will result in additional current consumption from the respective VXI backplane power supply.
Typical Time Allowed to Open/Close a Channel	30 msec
Interface Cables	Two 2-meter long twisted pair ribbon mating cables are supplied. Option 001 deletes these cables.
Module Size/Device Type	B, register-based
Connectors Used	P1 and P2 (optional)
Number of Slots	1
VXIbus Interface Capability	Interrupter, D16
Interrupt Level	1 - 7, selectable
Power Requirements	Voltage: +5V +12V Peak Module Current [IPM (A)]: 0.10 0.10 Dynamic Module Current [IDM (A)]: 0.10 0.01 (Does not include power from internal supplies to external loads.)

Appendix B
Relay Driver Register Definitions

Types of Registers

The Relay Driver Module is a register-based slave device. There are 72 independent relay drivers on the card that are controlled using the Relay Driver registers. There are five register types on this module.

Register	Description
ID Register	Identifies Agilent Technologies as the manufacturer, and the card is an A16 register-based device.
Device Type Register	Identifies card as an Agilent E1339A or Agilent Z2309A.
Status/Control Register	When read, is used to return device specific status information. When written to, is used to set control bits.
Configuration Registers	These three registers are used to configure the module for different operating characteristics when used with the downloaded SCPI driver or with the VXI*plug&play* driver.
Channel Enable Registers	These six registers control the state of the relay drivers on the module.

Note If SCPI or VXI*plug&play* is used to control this module, register programming is not recommended, since these drivers maintain an image of the card state. If the card state is altered by using register writes, these drivers will be unaware of these changes.

Addressing the Registers

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

- **Register Access with Logical Address**: This method uses the logical address directly to access a particular card using VXI:READ and VXI:WRITE commands through a command module.
- **Register Access with Memory Mapping:** This method can be used with an embedded controller that locates A16 data space within its memory map. The memory mapping allows registers to be directly read or written with moves to/from memory.

Register Access with Logical Address

When using the Agilent E1406 Command Module or E1300B/01B Mainframe to access registers via VXI:READ and VXI:WRITE commands, the logical address is used to determine which VXI module is being accessed. The factory setting of the Relay Driver Logical Address switch is 120 (78_h).

Refer to the Agilent E1406 Command Module or the E1300B/01B Mainframe documentation for use of the VXI:READ and VXI:WRITE commands and other related commands.

Example: Accessing the Status/Control Register

This example shows the Status/Control Register being accessed from an external or embedded controller via GPIB. Commands are sent to the Agilent E1406 Command Module

VXI:WRITE 120,4,#HFFFF	*Writes FFFF $_h$ to Control Register*
VXI:READ? 120,4	*Reads from Status Register*

Register Access with Memory Mapping

When using an embedded controller, VXI A16 address space is usually mapped to some block of memory within the controller's addressable memory space. See your embedded controller's manual to determine where VXI A16 is mapped, as there may be other methods of accessing the VXI backplane. This section shows the way A16 addresses are calculated for a module.

In the Command Module A16 address space starts at $1F0000_h$ ($_h$ = HEX). The A16 space is divided so that modules are addressed only at locations beginning with $C000_h$ within A16. Each module is allocated 64 register addresses (40_h). The module base address is related to the logical address set by the logical address switch on the module:

$$\textbf{base address } (_h) = (\text{logical address})_h * 40_h + C000_h$$

For the Relay Driver module, the factory-set logical address is 120 (78_h), so to address the Status/Control register of the module using the Command Module, use:

$$\textbf{base address} = (78_h)*(40_h) + C000_h = DE00_h$$

$$\textbf{register address} = [\text{A16 location}]_h + [\text{base address}]_h + [\text{register offset}]_h$$
$$\textbf{register address} = 1F0000_h + DE00_h + 04_h = 1FDE04_h$$

Reading from the Registers

You can read the following Relay Driver registers. A description of each register follows. See "Register Definitions" for additional details on each register.

Register	Address
ID Register	base + 00_h
Device Type Register	base + 02_h
Status/Control Register	base + 04_h
Configuration Register 0	base + 06_h
Configuration Register 1	base + 08_h
Configuration Register 2	base + $0A_h$
Channel Enable Register 0	base + 10_h
Channel Enable Register 1	base + 12_h
Channel Enable Register 2	base + 14_h
Channel Enable Register 3	base + 16_h
Channel Enable Register 4	base + 18_h
Channel Enable Register 5	base + $1A_h$

ID Register

For the Relay Driver module, a read of the ID register (base address + 00_h) returns $FFFF_h$ since the relay driver is manufactured by Agilent Technologies and is an A16 only, register-based device.

Device Type Register

For the Relay Driver module, a read of the Device Type register (base address + 02_h) returns 0181_h. This indicates it is a model Agilent Z2309A or Agilent E1339A.

Status/Control Register

A 30 msec timer on the module is set to "busy" whenever a Relay Driver register is written to, including opens or closes. A read of the Status/Control register (base + 04_h) returns a '1" in bit 7 when the module is not busy or returns a '0" in bit 7 when the module is busy.

An interrupt is generated after any of the Relay Driver registers are written. Bit 6 of the Status Register is used to enable/disable interrupts from the card. If bit 6 is returned as a "0" interrupts are enabled. If bit 6 is returned as a "1", interrupts are disabled.

Bit 14 is the MODID bit. When a "0" is returned in bit 14, the module has been selected with a high state on the P2 MODID line. If a 'I" is returned, the module has not been selected.

Configuration Registers

A read of the configuration registers will return their current value. All bits are normally left open which will return $FFFF_h$. There are 24 bits of input contained in the 8 least significant bits used in the three Configuration Registers. The upper 8 bits of each configuration register are always high. The only bits defined are bits CNFG 0 through CNFG 4 in Configuration Register 0. All other bits are ignored by the SCPI Driver.

Channel Enable Registers

A read of any of the Channel Enable registers always returns $FFFF_h$, regardless of the driver states.

Writing to the Registers

You can write to the following Relay Driver Module registers. A description of each register follows. See "Register Definitions" for additional details on each register

Register	Address
Status/Control Register	base + 04_h
Channel Enable Register 0	base + 10_h
Channel Enable Register 1	base + 12_h
Channel Enable Register 2	base + 14_h
Channel Enable Register 3	base + 16_h
Channel Enable Register 4	base + 18_h
Channel Enable Register 5	base + $1A_h$

Status/Control Register

The Status/Control register (base + 04_h) is set to bit 0. Writing a "1" to bit 0 resets the module. All drivers will be open, no longer sinking current. Resetting the module will also enable interrupts. It is necessary to write a "0' to bit 0 after the reset has been performed before any other commands can be programmed and executed.

To disable the interrupt generated when channels are opened/closed, write a "1' to bit 6 of the Status/Control Register.Typically, interrupts are disabled when doing register level access to a module. Refer to the operating manual of the Command Module or the embedded controller being used to handle interrupts. A SYSRESET will re-enable interrupts.

Channel Enable Registers

Writes to the Channel Enable registers (base + 10_h through base + lA_h) enable you to open or close the desired channel (see "Register Definitions"). When a channel is closed, it sinks current. Setting a bit to "1" closes a channel. Writing a "0" to a channel opens the channel, thus causing it to stop sinking current.

Register Definitions

ID Register

ID Register (base + 00_h)																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	Manufacturer ID*															

* Returns $FFFF_h$ = Agilent Technologies A16 only register-based device

Device Type Register

Device Type Register (base + 02_h)																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined															
Read	0181_h															

Status/Control Register

The Status/Control register provides status information for the system. The meanings of the bits applicable to the Relay Driver are:

- A = Reset to power-on state (all channels open) by writing a "1" in this bit (must be set back to "0")
- B = Disable interrupt by writing a "1" in this bit (set back to "0" with a reset)
- C = Undefined
- D = Status "busy" = "0", "not busy" = "1"
- E = Status inderrupt disable = "1", interrupt enable = "0"
- F = MODID bit = "0" module has been selected

Status/Control Register (base + 04_h)																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Write	Undefined									B	Undefined					A
Read	C	F	Undefined						D	E	Undefined					

Configuration Registers

The configuration registers set the configuration of the Relay Driver Module, such as 72-channel or 36-channel mode. Only CNFG 0 through CNFG 4 on Configuration Register 0 are defined. All other CNFG bits are ignored. CNFG 0 through CNFG 4 return "1" for unconnected inputs and return "0" for inputs connected to ground.

In addition, these bits are read from the hardware pins only when the driver is started, such as at power-on when using a command module with the downloaded driver. The meanings for CNFG 0 through 4 are:

- CNFG 0 returns "1" = 72-channel, "0" = 36-channel
- CNFG 1 returns "1" = drivers actuated simultaneously, "0" = channels opened/closed one at a time (when CNFG 0 is connected to GROUND)
- CNFG 2 returns "1" = continuous operation, "0" = pulsed operation (when CNFG 0 is connected to ground).
- CNFG 3 returns "1" = self-test timer busy bit check at 30 msec, "0" = self-test timer busy bit check at 50 msec.
- CNFG 4 returns "1" = self-test timer busy bit check at 30 msec or 50 msec (depends on CNFG 3 setting), "0" = no self-test timer busy bit check

Configuration Register 0 (base + 06_h)									
	15 - 8	7	6	5	4	3	2	1	0
Write	No Effect								
Read	FF_h	CNFG 7	CNFG 6	CNFG 5	CNFG 4	CNFG 3	CNFG 2	CNFG 1	CNFG 0

Configuration Register 1 (base + 08_h)									
	15 - 8	7	6	5	4	3	2	1	0
Write	No Effect								
Read	FFh	CNFG 15	CNFG 14	CNFG 13	CNFG 12	CNFG 11	CNFG 10	CNFG 9	CNFG 8

Configuration Register 2 (base + $0A_h$)									
	15 - 8	7	6	5	4	3	2	1	0
Write	No Effect								
Read	FFh	CNFG 23	CNFG 22	CNFG 21	CNFG 20	CNFG 19	CNFG 18	CNFG 17	CNFG 16

Channel Enable Registers

In 72-channel mode, each channel is addressed individually. In 36-channel mode, the channels are paired Ch0/Ch1, Ch2/Ch3, etc.l

Channel Enable Register 0 (Channels 0 - 11) (base + 10_h)													
	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
Write	N/A	Ch11	Ch10	Ch9	Ch8	Ch7	Ch6	Ch5	Ch4	Ch3	Ch2	Ch1	Ch0
Read	Always Returns $FFFF_h$												

Channel Enable Register 1 (Channels 12 - 23) (base + 12_h)													
	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
Write	N/A	Ch23	Ch22	Ch21	Ch20	Ch19	Ch18	Ch17	Ch16	Ch15	Ch14	Ch13	Ch12
Read	Always Returns $FFFF_h$												

Channel Enable Register 2 (Channels 24 - 35) (base + 14_h)													
	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
Write	N/A	Ch35	Ch34	Ch33	Ch32	Ch31	Ch30	Ch29	Ch28	Ch27	Ch26	Ch25	Ch24
Read	Always Returns $FFFF_h$												

Channel Enable Register 3 (Channels 36 - 47) (base + 16_h)													
	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
Write	N/A	Ch47	Ch46	Ch45	Ch44	Ch43	Ch42	Ch41	Ch40	Ch39	Ch38	Ch37	Ch36
Read	Always Returns $FFFF_h$												

Channel Enable Register 4 (Channels 48 - 59) (base + 18_h)													
	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
Write	N/A	Ch59	Ch58	Ch57	Ch56	Ch55	Ch54	Ch53	Ch52	Ch51	Ch50	Ch49	Ch48
Read	Always Returns $FFFF_h$												

Channel Enable Register 5 (Channels 60 - 71) (base + $1A_h$)													
	15 - 12	11	10	9	8	7	6	5	4	3	2	1	0
Write	N/A	Ch71	Ch70	Ch69	Ch68	Ch67	Ch66	Ch65	Ch64	Ch63	Ch62	Ch61	Ch60
Read	Always Returns $FFFF_h$												

Appendix C
Relay Driver Error Messages

Error Types

This appendix lists the error messages generated by the Agilent E1339A Relay Driver module firmware when programmed by SCPI. Errors with negative values are governed by the SCPI standard. Error messages with positive values are not governed by the SCPI standard. The definitions for error messages with negative numbers are:

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

Error Messages

The following table lists all of the positive-valued error messages and *some* of the negative-valued error messages associated with the Relay Driver. See the *Agilent E1306 Command Module User's Manual* or the *Agilent E1406 Command Module User's Manual* for further information or description
of negative-valued errors that are not listed in the table..

Code	Error Message	Potential Cause(s)
-109	Missing parameter	A required parameter in a command is missing.
-141	Illegal character data	Attempting to execute a command with a character not applicable to the command.
-211	Trigger Ignored	Trigger received when scan not enabled, or trigger received after scan complete, or trigger too fast.
-213	INIT Ignored	Attempting to execute an INIT command when a scan is already in progress.
-222	Data out of range	Parameter value is outside valid range.
-224	Illegal parameter value	Attempting to execute a command with a parameter not applicable to the command.
-240	Hardware Error	Command failed due to a hardware problem.
-241	Hardware Missing	E1339A SCPI driver is not downloaded
-310	System Error	Internal driver error. This error can result if an excessively long parameter list is entered.

Code	Error Message	Potential Cause(s)
1500	External trigger source already allocated	Assigning an external trigger source to a switchbox when the trigger source has already been assigned to another instrument.
1510	Trigger source non-existent	Selected trigger source is not available on this platform (e.g., some triggers are not available on the Agilent E1300/E1301 VXI B-size mainframes).
2000	Invalid card number	Addressing a module (card) in a swtchbox 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 Relay Driver module).
2006	Command not supported on this card	Sending a command to a module (card) in a switchbox than is not supported by this card.
2008	Scan list not initialized	Executing an INIT before sending the SCAN command.
2009	Too many channels in channel list	Attempting to address more channels than available in the switchbox *channel_list*.
2010	Scan mode not allowed on this card	The selected scanning mode is not allowed with this module or you have misspelled the mode parameter (see SCAN:MODE command).
2011	Empty channel list	No valid channels are specified in the *channel_list*.
2012	Invalid channel range	Invalid channel(s) specified in SCAN <*channel_list*> command. Attempting to begin scanning when no valid channel list is defined.
2600	Function not supported on this card	Sending a command to a module (card) in a switchbox that is not supported by this card.
2601	Channel list required	Sending a command requiring a *channel_list* without the *channel_list*.

Index

A

ABORT Subsystem
ABORt, 60
Agilent E1339A Switchbox Installation, 28
ARM Subsystem
ARM:COUNt <*number*> MIN | MAX, 62
ARM:COUNt? [MIN | MAX], 63

B

base address, 96

C

Certification, 9
Channel Enable Registers, 98, 101
Command Module, 13, 17
Command Types
Abbreviated Commands, 58
Command Separator, 58
Common Command Format, 57
Implied Commands, 58
Linking Commands, 59
Parameters, 58
SCPI Command Format, 57
Configuration
Changing Timer Delay, 27
Connecting User Inputs, 22
Internal Power Supply Voltage, Setting, 25
Interrupt Priority Jumpers, Setting, 24
IRO jumper setting, 25
Logical Address Switch, Setting, 25
Pin Definitions - 36-Channel (P200), 23
Pin Definitions - 72- Channel (P200), 24
Pin Definitions - 72-Channel (P400), 24
Pin Definitions -36-Channel (P400), 23
Setting Pulsed/Continuous Operation, 27
Setting Simultaneous/Stepped Actuation, 27
Configuration Registers, 98, 100
Connecting User Inputs, 22

D

Declaration of Conformity, 11
Device Type Register, 97, 99
DISPlay Subsystem
DISPlay:MONitor:CARD <*number*> |AUTO, 64
DISPlay:MONitor:CARD?, 64
DISPlay:MONitor[:STATe] <*mode*>, 65
DISPlay:MONitor[:STATe]?, 65
Downloading the SCPI Driver, 29

E

Error Messages, 103
Error Query Response (SYST
ERR?), 51
Error Types, 103
Example Programs
Accessing the Status/Control Register, 96
Advancing Scan Using TRIGger, 86
Channel Lists/Ranges (36-Channel Mode), 19
Channel Lists/Ranges (72-Channel Mode), 19
Closing a Channel (36-Channel Mode), 40
Closing a Channel (72-Channel Mode), 38
Closing Relay Driver Module Channels, 75
Enabling "Trig Out" Port, 71
Enabling Continuous Scanning, 67
Enabling ECL Trigger Bus Line 0, 70
Enabling Status Byte Register Bit 8, 81
Enabling the Monitor Mode, 65
Enabling TTL Trigger Bus Line 7, 72
Opening Switchbox Channels, 76
Query "Trig Out" Port State, 71
Query Channel Closure, 76
Query Channel Open State, 77
Query Continuous Scanning State, 67
Query ECL Trigger Bus Enable State, 70
Query Number of Scans, 63
Query the Operation Status Enable Register, 81
Query Trigger Source, 89
Query TTL Trigger Bus Enable State, 73
Reading Card #1 Description, 83
Reading Card #1 Model Number, 84
Reading the Error Queue, 85
Reading the Operation Status Register, 82
Reset, Self-Test, and Module ID, 29
Scan Complete Interrupt, 55

Example Programs (cont'd)
Scanning Channels (36-Channel Mode), 44
Scanning Channels (72-Channel Mode), 42
Scanning Using Bus Trigger, 89
Scanning Using External Devices, 78
Scanning Using External Triggers, 88
Scanning Using Trig Out/Event In Ports, 51
Select Switchbox Module for Monitoring, 64
Setting Card to its Power-On State, 84
Setting Ten Scanning Cycles, 62
Starting a Single Scan, 68
Stopping a Scan With ABORt, 61
Three-Module Switchbox, 18
Typical Switchbox Instrument, 17
Example Programs, Using, 37

I

ID Register, 97, 99
IEEE 488.2 Common Commands, 91
INITiate Subsystem
INITiate:CONTinuous <*mode*>, 66
INITiate:CONTinuous?, 67
INITiate[:IMMediate], 67
Installation
Agilent E1339/Z2309A Switchbox Installation, 28
Agilent E1339A Switchbox Installation, 28
Downloading the SCPI Driver, 29
Installing Agilent VXI*plug&play* Drivers, 29
Typical Software Configuration, 29
ZSWITCHBOX Driver Considerations, 29
Installing Agilent VXI*plug&play* Drivers, 29
Internal Power Supply Voltage, Setting, 25
Interrupt Priority Jumpers, Setting, 24
IRQ jumper setting, changing, 25

L

Logical Address Switch, Setting, 25

M

MODID bit, 97

O

Operation
Modes of Operation, 15
Operations Overview, 26
Setting 36/72-Channel Operation, 26
Using 36-Channel Mode, 16
Pulsed/Continuous Output, 16
Simultaneous/Stepped Actuations, 16
Using 72-Channel Mode, 16
OUTPut Subsystem
OUTPut:ECLT*n*[:STATe] <*mode*>, 69
OUTPut:ECLT*n*[:STATe]?, 70
OUTPut[:EXTernal][:STATe], 70
OUTPut[:EXTernal][:STATe]?, 71
OUTPut:TTLTrg*n*[:STATe] <*mode*>, 72
OUTPut:TTLTrg*n*[:STATe]?, 73

R

Reading Registers
Channel Enable Registers, 98
Configuration Registers, 98
Device Type Register, 97
ID Register, 97
Status/Control Register, 97
Recalling States (*RCL), 50
register address, 96
Register Addressing
With Logical Address, 96
With Memory Mapping, 96
Register Definitions
Channel Enable Registers, 101
Configuration Registers, 100
Device Type Register, 99
ID Register, 99
Status/Control Register, 99
Relay Driver Module
Configuration, 22
Description, 13
Error Messages, 103
Programming, 17
Register Definitions, 95
Specifications, 93
Response to Reset (*RST), 51

ROUTe Subsystem
[ROUTe:]CLOSe <*channel_list*>, 74
[ROUTe:]CLOSe? <*channel_list*>, 75
[ROUTe:]OPEN <*channel_list*>, 76
[ROUTe:]OPEN? <*channel_list*>, 77
[ROUTe:]SCAN <*channel_list*>, 77

S

Safety Symbols, 10
Scan Mode Operation Commands, 48
Scanning Channels
Scanning Trigger Sources, 49
Scanning With External Instruments, 50
Setting Scan Mode Operation, 47, 48
Using Scanning Trigger Sources, 47
Using the Scan Complete Bit, 50
Scanning Trigger Sources, 49
Scanning With External Instruments, 50
SCPI Command Quick Reference, 90
SCPI Command Reference, 59
Setting Scan Mode Operation, 47
Specifying SCPI Commands, 18
STATus Subsystem
STATus:OPERation:CONDition?, 81
STATus:OPERation:ENABle <*unmask*>, 81
STATus:OPERation:ENABle?, 81
STATus:OPERation[:EVENt]?, 82
STATus:PRESet, 82
Status/Control Register, 97, 98, 99
Storing States (*SAV), 50
Switch Card Numbers, 18
Switch Channel Addresses, 18
Switch Channel Numbers, 18
Switchbox Definition, 17
SYSTem Subsystem
SYSTem:CDEScription? <*number*>, 83
SYSTem:CPON <*number*> | ALL, 83
SYSTem:CTYPE? <*number*>, 84
SYSTem:ERRor?, 85

T

TRIGger Subsystem
TRIGger[:IMMediate], 86
TRIGger:SOURce <*source*>, 87
TRIGger:SOURce?, 89
Types of Registers, 95
Typical Software Configuration, 29

U

Using 36-Channel Mode
Operations Overview, 33
Scanning Channel Pairs, 35
Setting Pulsed/Continuous Outputs, 34
Setting Simultaneous/Stepped Actuations, 34
Switching Channel Pairs, 34
Using 72-Channel Mode
Operations Overview, 32
Scanning Channels, 32
Switching Channels, 32
Using Scanning Trigger Sources, 47
Using the Scan Complete Bit, 50

V

Verifying Initial Operation, 29

W

WARNINGS, 10
Warnings and Cautions, 21
Warranty, 9
Writing to Registers
Channel Enable Registers, 98
Status/Control Register, 98

Z

ZSWITCHBOX Driver Considerations, 29