Programmer Manual

Tektronix

PS2520G & PS2521G
Programmable Power Supplies

070-9197-00

Printed in the U.S.A.

Tektronix, Inc., P.O. Box 1000, Wilsonville, OR 97070–1000

WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; or c) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THIS PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESSED OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

List of Figures

List of Tables

Preface

This manual explains how to use and program the PS2520G and PS2521G Programmable Power Supplies over the General Purpose Interface Bus (GPIB). The following sections make up the body of this manual:

- *Getting Started* describes how to set up the power supply and GPIB systems.
- *Syntax and Commands* describes the structure of messages your program sends to the power supply.
- *Status and Events* describes how to use the event messages in your programs.

For more information about the GPIB, refer to the standards IEEE 488.1-1987, IEEE 488.2-1992, and SCPI-1994 (Standard Commands for Programmable Instruments).

Refer to the *PS2520, PS2520G, PS2521 & PS2521G User Manual* (070-9196-XX) for product specifications, safety summary, operating and service information, and a list of standard accessories for the programmable power supplies.

Getting Started

With a computer (controller), you can operate the PS2520G and PS2521G Programmable Power Supplies over the GPIB. This section explains how to perform the following tasks:

- Set up GPIB systems
- Connect the programmable power supply to a GPIB controller
- Set GPIB address of the programmable power supply
- Test the GPIB connection

Setting Up the GPIB System

Observe these rules when you set up the programmable power supply with a GPIB system:

- Each device on the bus needs a unique device address. No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 m (6 ft) of cable used.
- Do not use more than 20 m (65 ft) of cable to connect devices to a bus.
- Turn on at least two-thirds of the devices on the GPIB system while you use the system.
- Configure the devices on the system as shown in Figure 1. Do not use loop or parallel configurations.

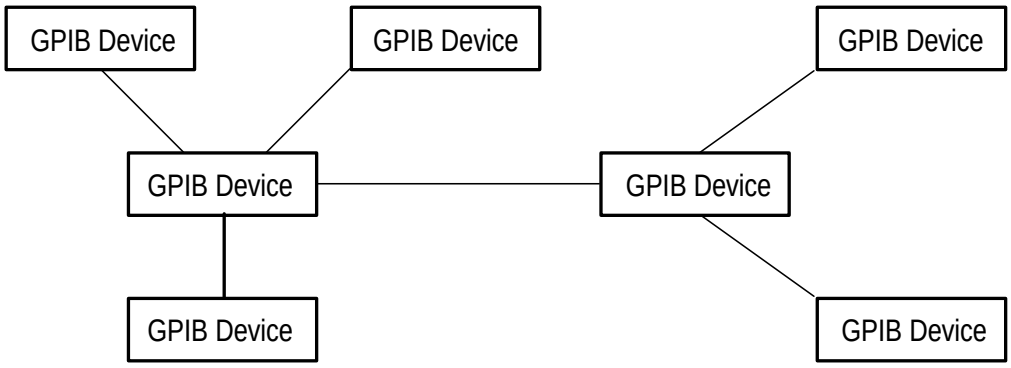

Figure 1: Typical GPIB Network Configuration

Connecting a Controller

You must have a GPIB controller, such as a PC with a GPIB card, to operate the programmable power supplies over the GPIB interface.

Figure 2 on page 3 shows the location of the GPIB port on the rear panel.

Connect the programmable power supply to a GPIB controller as follows:

1. Connect one end of a GPIB cable to the GPIB controller.
2. Connect the other end of the GPIB cable to the GPIB port on the programmable power supply.
3. Turn on the programmable power supply.
4. Turn on the GPIB controller.

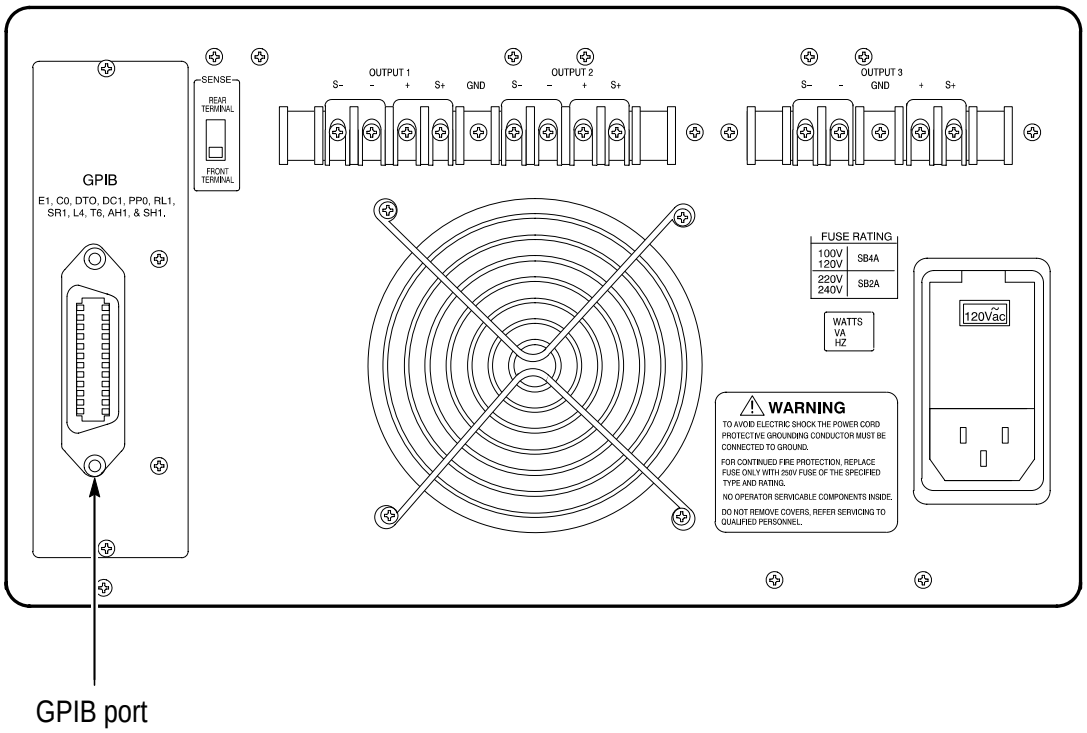

Figure 2: GPIB Port

Setting the Address

Change the GPIB address of the programmable power supply using the following procedure:

NOTE. *Each device connected to the same controller must have a unique GPIB address. The factory default address for the PS2520G and PS2521G Programmable Power Supplies is 12.*

---

1. Make sure the AUTO SEQ function is off. (The indicator for AUTO must be off).
2. Press **LOCAL** → (number) → (return ◄┘) to enter the GPIB address of the power supply. (The address must be a number between 0 and 30.)

3. To check the address setting of the power supply, press the **LOCAL** button again. The display remains active for about 3 seconds to allow you to view or change the setting.

Testing the GPIB Connection

To test whether the GPIB connection is working, send a GPIB query from the computer. For example, the query

```
*idn?
```

should return the name of the instrument, SCPI version, and firmware version in the following form:

```
TEKTRONIX,<model>,0,SCPI:<year> FW<version>
```

If you do not get a proper response from the programmable power supply, check to make sure the power is on, all cable connections are secure, and the GPIB address is correct.

Syntax and Commands

This section provides an overview of the commands for the PS2520G and PS2521G Programmable Power Supplies. This section includes the following topics:

- A brief introduction to SCPI
- A description of the command syntax
- Instructions on how to enter commands
- A summary of commands by functional group

In addition, the section *Command Descriptions* on page 19 lists the commands alphabetically and provides a detailed description for each command.

The commands for the programmable power supplies are compatible with IEEE-488.1, IEEE-488.2, and SCPI-1994.0 standards.

SCPI

SCPI (Standard Commands for Programmable Instruments) is a standard created by an international consortium of the major manufacturers of test and measurement equipment. SCPI uses IEEE-488.2 syntax to provide common commands for the identical functions of various programmable instruments.

The standard simplifies the task of programming a group of instruments that use SCPI. Instead of having to learn different commands for every instrument, the programmer may use the same commands for the identical functions of each instrument.

Commands and Queries

The controller sends instructions to the instrument in the form of commands or queries. Commands modify control settings or tell the instrument to perform a specific action. Queries cause the instrument to send data or status information back to the controller. A question mark at the end of a command identifies it as a query.

Different product manuals may also use the terms “program messages” or simply “commands” to refer to commands and queries as a whole. This manual uses the term “commands” to mean any type of instruction from the controller to the programmable instrument.

Command Syntax

Any instruction that you send to an instrument that complies with SCPI must have at least three basic elements:

- Command header
- Parameter (if required)
- Message terminator or separator

Command Header

The command header has a hierarchical structure that can be represented by a command tree (Figure 3). A mnemonic designates each level of the hierarchy. A colon separates the levels.

The top level of the tree is the root level. A root node is a mnemonic at the root level. A root node and one or more lower-level nodes form a header path to the last node called the leaf node.

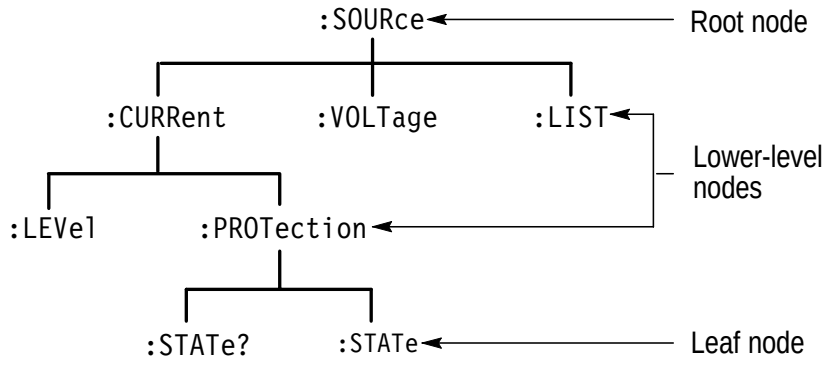

Figure 3: Tree Hierarchy

The header path and leaf node together form the command header. Figure 4 shows the command header for the leaf node indicated in Figure 3.

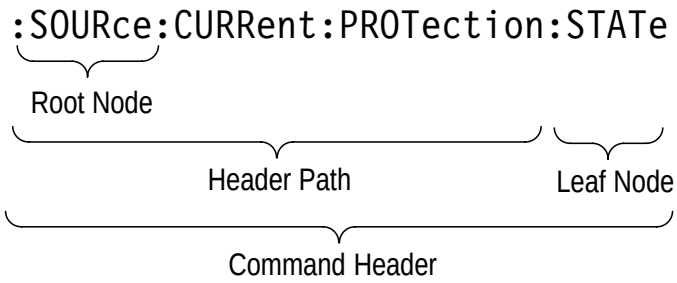

Figure 4: Command Header

Parameter

You must include values for commands that have parameters. In this manual, the < > symbols enclose the parameter type when stating the syntax of the command. For example, the syntax of the command in Figure 5 includes the Boolean parameter type.

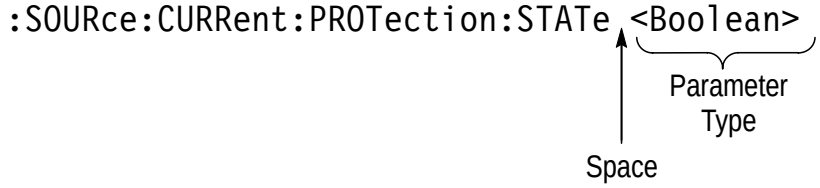

Figure 5: Command Header with Parameter

Table 1 defines the Boolean and other parameter types for the programmable power supplies.

Table 1: Parameter Types for Syntax Descriptions

Parameter Type	Description	Example
Boolean	Boolean numbers or values	ON or 1 OFF or 0
NR1	Integers	0, 1, 15
NR2	Decimal numbers	1.2, 3.141516, 6.5
NR3	Floating point numbers	3.1E–1, 2.01E+1
NRf	NR1, NR2, or NR3	1, 1.2, 3.1E–1
string	Alphanumeric characters (must be within quotation marks)	"No error"
MAXimum|MINimum	Special mnemonics for setting the parameter to the largest or smallest value the instrument allows.	MAX, MIN

For the actual value of the parameter type <Boolean>, you may enter `0` or `OFF` or you may enter `1` or `ON`.

For the command in Figure 5, entering `0` or `OFF` turns the overcurrent protection (OCP) off and entering `1` or `ON` turns the OCP on.

The following example includes both the header and a value for the parameter type:

```
SOURce:CURRent:PROTection:STATe OFF
```

Parameter values that appear in this manual are often separated by a vertical line. This vertical line means the same thing as the word "or." For example, values for the parameter <Boolean> are

```
0|1|OFF|ON
```

This is the same thing as saying "0 or 1 or OFF or ON." Any single value is a valid parameter.

NOTE. *Do not include the* `<`, `>`, `or` `|` *symbols when entering the actual value for a parameter.*

Message Terminator and Message Separator

In accordance with IEEE 488.2-1992, the accept any of the following message terminators:

- LF^END Line feed code (hexadecimal 0A) with END message
- LF Line feed code
- <dab>^END Last data byte with END message

These terminators are compatible with most controller application programs.

A semicolon separates one command from another when the commands appear on the same line.

Entering Commands

The standards that govern the command set for the programmable power supplies allow for a certain amount of flexibility when you enter commands. For example, you can abbreviate many commands or combine commands into one message that you send to the programmable power supply. This flexibility, called friendly listening, saves programming time and makes the command set easier to remember and use.

Command Characters

The programmable power supplies are not sensitive to the case of command characters. You can enter commands in either uppercase or lowercase.

You can precede any command with white space characters. White space characters include a space or any combination of the ASCII control characters hexadecimal 00 through 1F except for the character 0A (line feed). You must, however, use at least one space between the parameter and the command header.

Abbreviating Commands

Most commands have a long form and a short form. The listing for each command in this section shows the abbreviations in upper case. For example, you can enter the query `SOURce:VOLTage?` simply as `SOUR:VOLT?` (or `sour:volt?`).

The brackets around a mnemonic indicate that the programmable power supply assumes this level of the command header by default. It is not necessary, therefore, for this mnemonic to appear as part of the header when you send the command. For example, you can abbreviate the command

```
[SOURce:]VOLTage[:LEVel][:IMMediate][AMPLitude] 10
```

to

```
volt 10
```

Because the programmable power supply assumes that a command starts from the root, you have the option of beginning the initial command header with a colon (:).

Combining Commands

You can combine (concatenate) commands and queries using a semicolon (;). The programmable power supply executes concatenated commands in the order it receives them. When you concatenate queries, the programmable power supply combines the responses into a single response message. For example, if the current and voltage limits are set to 1 A and 20 V, the command

```
curr?;volt?
```

returns the message

```
0.100E+1;0.200E+2
```

If the command that follows the semicolon has a different header path from the root level, you must use a colon to force a return to the root level:

```
MEASure:CURRent?;:OUTPut:STATe?
```

If the command that follows the semicolon has the same header path, you may omit the colon and the path and state only the new leaf

node. This makes it possible, for example, to shorten the concatenated query

```
:MEASure:CURRent?;:MEASure:VOLTage?
```

into

```
MEASure:CURRent?;VOLTage?
```

You can combine commands and queries into the same message. Note, for example, the following combination:

```
sour:volt 10;:sour:volt?
```

or

```
sour:volt 10;volt?
```

Summary of Commands

The tables in this section summarize the command set of the programmable power supplies. These tables divide the commands into the following functional groups:

- General Setting Commands
- Status Commands
- Miscellaneous Commands

The tables also provide a brief description of each command.

General Setting Commands

Table 2 lists the general setting commands that control and query the settings of the power supply. To a large extent, the commands duplicate the function of the front panel controls and indicators.

Table 2: General Setting Commands

Command	Description
`INSTrument:NSELect 1|2|3`	Selects output 1, 2, or 3.
`INSTrument:NSELect?`	Returns selected output.
`INSTrument[:SELect] <identifier> OUT1|OUT2|OUT3`	Selects output 1, 2, or 3.
`INSTrument[:SELect]?`	Returns selected output.
`INSTrument:COUPle:TRACking?`	Returns selected independent, parallel-tracking, or series-tracking mode.
`INSTrument:COUPle:TRACking NONE|PARallel|SERies`	Selects independent, parallel-tracking, or series-tracking mode.
`MEASure[:SCALar]:CURRent[:DC]?`	Returns actual output current.
`MEASure[:SCALar]:VOLTage[:DC]?`	Returns actual output voltage.
`OUTPut:PROTection:CLEar`	Clears overvoltage and overcurrent protection errors.
`OUTPut[:STATe] <Boolean>`	Sets the output state on or off.
`OUTPut[:STATe]?`	Returns the output state (on or off).

Table 2: General Setting Commands (Cont.)

Command	Description
[SOURce:]CURRent[:LEVel] [:IMMediate][:AMPLitude] <NRf>|MAX|MIN	Sets the current limit.
[SOURce:]CURRent[:LEVel] [:IMMediate][:AMPLitude]?	Returns the current-limit setting.
[SOURce:]CURRent :PROTection:STATe <Boolean>	Sets the Overcurrent Protection (OCP) on or off.
[SOURce:]CURRent :PROTection:STATe?	Returns the state of the Overcurrent Protection (OCP) setting as either on or off.
[SOURce:]VOLTage[:LEVel] [:IMMediate][:AMPLitude] <NRf>|MAX|MIN	Sets the voltage limit.
[SOURce:]VOLTage[:LEVel] [:IMMediate][:AMPLitude]?	Returns the voltage limit setting.
[SOURce:]VOLTage :PROTection[:LEVel] <NRf>|MAX|MIN	Sets the overvoltage protection (OVP) level.
[SOURce:]VOLTage :PROTection[:LEVel]?	Returns the overvoltage protection (OVP) setting.

Status Commands

Table 3 lists the status commands that set and query the various registers and queues that make up the status and event structure of the programmable power supplies.

Table 3: Status Commands

Command	Description
*CLS	Clears the status data structures.
*ESE <NRf>	Sets the Event Status Enable Register (ESER).
*ESE?	Returns contents of Event Status Enable Register (ESER).

Table 3: Status Commands (Cont.)

Command	Description
*ESR?	Returns and clears the contents of Standard Event Status Register (SESR).
STATus:OPERation:CONDition?	Returns the contents of the OPERation condition register. Returns NR1.
STATus:OPERation:ENABle <NRf>	Sets the contents of the enable mask for the OPERation event register.
STATus:OPERation:ENABle?	Returns the contents of the enable mask for the OPERation event register. Returns NR1.
STATus:OPERation[:EVENt]?	Returns and clears the contents of the OPERation event register.
STATus:OPERation:INSTrument:CONDition?	Returns the contents of the OPERation INSTrument condition register. Returns NR1.
STATus:OPERation:INSTrument:ENABle <NRf>	Sets the contents of the enable mask for the OPERation INSTrument event register.
STATus:OPERation:INSTrument:ENABle?	Returns the contents of the enable mask for the OPERation INSTrument event register. Returns NR1.
STATus:OPERation:INSTrument[:EVENt]?	Returns and clears the contents of the OPERation INSTrument event register.
STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?	Returns the contents of the OPERation INSTrument ISUMmary<n> condition register (ISUMmary1, ISUMmary2, or ISUMmary3). Returns NR1.
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle <NRf>	Sets the contents of the enable mask for the OPERation INSTrument ISUMmary<n> event register (ISUMmary1, ISUMmary2, or ISUMmary3).
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle?	Returns the contents of the enable mask for the OPERation INSTrument ISUMmary<n> event register (ISUMmary1, ISUMmary2, or ISUMmary3). Returns NR1.

Table 3: Status Commands (Cont.)

Command	Description
STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?	Returns and clears the contents of the OPERation INSTrument ISUMmary<n> event register (ISUMmary1, ISUMmary2, or ISUMmary3).
STATus:PRESet	Presets the OPERation and QUEStionable status registers.
SYSTem:ERRor?	Reads the next item from the error/event queue.
STATus:QUEue[:NEXT]?	Reads the next item from the error/event queue (identical to SYSTem:ERRor?).
STATus:QUEStionable:CONDition?	Returns the contents of the QUEStionable condition register. Returns NR1.
STATus:QUEStionable:ENABle <NRf>	Sets the contents of the enable mask for the QUEStionable event register.
STATus:QUEStionable:ENABle?	Returns the contents of the enable mask for the QUEStionable event register. Returns NR1.
STATus:QUEStionable[:EVENt]?	Returns and clears the contents of the QUEStionable event register.
STATus:QUEStionable:INSTrument:CONDition?	Returns the contents of the QUEStionable INSTrument condition register. Returns NR1.
STATus:QUEStionable:INSTrument:ENABle <NRf>	Sets the contents of the enable mask for the QUEStionable INSTrument event register.
STATus:QUEStionable:INSTrument:ENABle?	Returns the contents of the enable mask for the QUEStionable INSTrument event register. Returns NR1.
STATus:QUEStionable:INSTrument[:EVENt]?	Returns and clears the contents of the QUEStionable INSTrument event register.
STATus:QUEStionable:INSTrument:ISUMmary<n>:CONDition?	Returns the contents of the QUEStionable INSTrument ISUMmary condition register (ISUMmary1, ISUMmary2, or ISUMmary3). Returns NR1.

Table 3: Status Commands (Cont.)

Command	Description
STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle <NRf>	Sets the contents of the enable mask for the QUEStionable INSTrument ISUMmary event register (ISUMmary1, ISUMmary2, or ISUMmary3).
STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle?	Returns the contents of the enable mask for the QUEStionable INSTrument ISUMmary event register (ISUMmary1, ISUMmary2, or ISUMmary3). Returns NR1.
STATus:QUEStionable:INSTrument:ISUMmary<n>[:EVENt]?	Returns and clears the contents of the QUEStionable INSTrument ISUMmary event register (ISUMmary1, ISUMmary2, or ISUMmary3).
*SRE <NRf>	Sets contents of Service Request Enable Register (SRER).
*SRE?	Returns contents of Service Request Enable Register (SRER).
*STB?	Reads Status Byte Register (SBR).

Miscellaneous Commands

Table 4 lists the miscellaneous commands that control general housekeeping functions of the programmable power supplies.

Table 4: Miscellaneous Commands

Command	Description
*IDN?	Returns instrument identification.
*OPC	Reports when operation is complete by setting the Operation Complete bit in SESR.
*OPC?	Reports when operation is complete. Same as *OPC except returns a 1 to the output queue and does not set the SESR bit.

Table 4: Miscellaneous Commands (Cont.)

Command	Description
*RST	Resets the protection levels and states, resets the current and voltage levels to zero, sets the output off, and sets memory point to 00.
*TST?	Initiates internal self-test and reports results.
*WAI	Wait to continue. This command forces sequential operation of commands. This command is required by IEEE 488.1-1987. The power supply, however, forces sequential operation of commands by design.
SYSTem:VERSion?	Returns the SCPI version level.
SYSTem:AUTO <Boolean>	Sets Auto Sequence on or off.[1]
SYSTem:AUTO?	Returns Auto Sequence mode.[1]

[1] **Not a SCPI standardized command.**

Command Descriptions

This section provides an alphabetical listing and a detailed description of each command. It also provides examples of each command and what the query form might return.

*CLS (No Query Form)

Clears the following status data structures:

- Standard Event Status Register
- Operation Event Status Register
- Questionable Event Status Registers
- Error/Event Queue

Syntax

```
*CLS
```

Examples

`*CLS` clears all event registers.

*ESE

Sets or returns the bits in the Event Status Enable Register (ESER).The ESER enables the Standard Event Status Register (SESR) to be summarized on bit 5 (ESB) of the Status Byte Register (SBR). Refer to Figure 12 on page 52 for an illustration of the ESER.

Syntax

```
*ESE <NRf>
*ESE?
```

Parameters

`<NR1>` is a number from 0 to 255. The binary bits of the ESER are set according to this value.

Returns

<NR1> is a number from 0 to 255 that indicates the decimal value of the binary bits of the ESER.

Examples

*ESE 48 sets the ESER to binary 0011 0000, which enables the CME and EXE bits.

*ESE? returns 129 if the ESER contains the binary value 1000 0001.

*ESR? (Query Only)

Returns and clears the contents of the Standard Event Status Register (SESR). Refer to Figure 11 on page 50 for an illustration of the SESR.

Syntax

*ESR?

Returns

<NR1> is a number from 0 to 255 that indicates the decimal value of the binary bits of the SESR.

Examples

*ESR? returns 160, if the SESR contains binary 1010 0000 (PON and CME bits set).

*IDN? (Query Only)

Returns the unique identification code of the power supply.

Syntax

*IDN?

Examples

*IDN? returns

TEKTRONIX,PS2521G,0,SCPI:94.0 FW:.10

INSTrument:NSELect

Selects Output 1, Output 2, or Output3. Only one output may be selected at a time for configuration.

Syntax

```
INSTrument:NSELect <NR1>
INSTrument:NSELect?
```

Parameters

1|2|3

Returns

1|2|3

Examples

INSTrument:NSELect 2 selects Output 2.
INSTrument:NSELect? returns 3 if Output 3 is selected.

INSTrument[:SELect]

Selects Output 1, Output 2, or Output3. An output must be selected before it can be configured and only one output may be selected at a time.

Syntax

```
INSTrument[:SELect] <identifier>
INSTrument[:SELect]?
```

Parameters

OUT1|OUT2|OUT3

Returns

OUT1|OUT2|OUT3

Examples

INSTrument[:SELect] OUT1 selects Output 1.
INSTrument[:SELect]? returns OUT2 if Output 2 is selected.

INSTrument:COUPle:TRACking

Selects independent, parallel-tracking, or series-tracking mode.

Syntax

```
INSTrument:COUPle:TRACking <identifier>
INSTrument:COUPle:TRACking?
```

Parameters

```
NONE|PARallel|SERies
```

NONE is independent mode
PARallel is parallel-tracking mode
SERies is series-tracking mode

Returns

```
NONE|PARallel|SERies
```

Examples

`INSTrument:COUPle:TRACking NONE` selects the independent mode.
`INSTrument:COUPle:TRACking?` returns SERies if series-tracking mode is selected.

MEASure[:SCALar]:CURRent[:DC]? (Query Only)

Returns actual output current.

Syntax

```
MEASure[:SCALar]:CURRent[:DC]?
```

Returns

```
<NR3>
```

Examples

`MEASure:CURRent?` might return `0.8E-2` to indicate that the load is drawing 0.008 A (8 mA).

MEASure[:SCALar]:VOLTage[:DC]? (Query Only)

Returns actual output voltage or sense input voltage.

Syntax

```
MEASure[:SCALar]:VOLTage[:DC]?
```

Returns

<NR3>

Examples

MEASure:VOLTage? might return 0.367E+1 to indicate the voltage at the output is 3.67 V.

*OPC

The command form (*OPC) sets the operation complete bit (bit 0) in the Standard Event Status Register (SESR) when all pending operations finish.

The query form (*OPC?) tells the programmable power supply to place an ASCII 1 in the Output Queue when the power supply completes all pending operations.

Syntax

```
*OPC
*OPC?
```

Returns

1

OUTPut:PROTection:CLEar (No Query Form)

Resets (clears) the Overvoltage Protection. This command duplicates the function of the OVP RESET button.

Syntax

```
OUTPut:PROTection:CLEar
```

OUTPut[:STATe]

Enables or disables the output of the power supply. ON (enabled) is signified by a 1 and OFF (disabled) by a 0. The query form returns a 1 if the output is on and a 0 if the output is off.

Syntax

```
OUTPut:STATe <Boolean>
OUTPut:STATe?
```

Parameters

0|1|OFF|ON

Returns

0|1

Examples

`OUTPut:STATe ON` enables the power supply output.

`OUTPut:STATe?` returns 1 if the power supply output is enabled.

*RST (No Query Form)

Resets the control settings of the programmable power supply to a set of known states but does not purge stored settings. Refer to Table 5 for the reset state of the control settings.

Table 5: State of Control Settings after *RST

Front Panel Control	Related Command	State
OUTPUT	OUTPut[:STATe]	OFF
CURRENT SET	[SOURce:]CURRent[:LEVel] [:IMMediate][:AMPLitude]	0
VOLTS SET	[SOURce:]VOLTage[:LEVel] [:IMMediate][:AMPLitude]	0

Table 5: State of Control Settings after *RST (Cont.)

Front Panel Control	Related Command	State
OCP ON/OFF	[SOURce:]CURRent:PROTection :STATe	ON
DELAY	—	0
AUTO SEQ	SYSTem:AUTO	OFF
RECALL (memory location)	—	00
OVP SET	[SOURce:]VOLTage:PROTection [:LEVel]	MAXimum (PS2520G = 38.5 V PS2521G = 22.5 V)
SERIES/INDEP PARA/INDEP	INSTrument:COUPle:TRACking	NONE

Syntax

```
*RST
```

[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]

Sets the current limit. The query form returns the current limit setting.

Syntax

```
[SOURce:]CURRent[:LEVel][:IMMediate]
[:AMPLitude] <NRf>|MAXimum|MINimum

[SOURce:]CURRent[:LEVel][:IMMediate]
[:AMPLitude]?
```

Parameters

On the PS2520G, <NRf> ranges from 0 to 1.500 (amps) for Output 1 and Output 2 and 0 to 3.000 (amps) for Output 3.

On the PS2521G <NRf> ranges from 0 to 2.500 (amps) for Output 1 and Output 2 and 0 to 5.000 (amps) for Output 3.

Returns

<NR3>

Examples

SOURce:CURRent 1.5 sets the current limit to 1.5 amps.

SOURce:CURRent? returns 0.1000E+1 if the current limit setting is 1 amp.

[SOURce:]CURRent:PROTection:STATe

Sets the overcurrent protection on or off. The query form returns the state of the overcurrent protection as either on or off.

Syntax

```
[SOURce:]CURRent:PROTect:STATe <Boolean>
[SOURce:]CURRent:PROTect:STATe?
```

Parameters

0|1|OFF|ON
OFF or 0 sets overcurrent protection off. ON or 1 sets overcurrent protection on.

Returns

0|1

Examples

SOURce:CURRent:PROTect:STATe OFF sets the overcurrent protection off.

SOURce:CURRent:PROTect:STATe? returns 1 if the overcurrent protection is on.

[SOURce:]VOLTage[:LEVel][:IMMediate][:AMPLitude]

Sets the voltage limit of the power supply. The query form returns the voltage limit setting.

Syntax

```
[SOURce:]VOLTage <NRf>|MAXimum|MINimum
[SOURce:]VOLTage?
```

Parameters

On the PS2520G, <NRf> is a positive integer or real number from 0 to 37.00 volts for Output 1 and Output 2 and 0 to 6.50 volts for Output 3.

On the PS2521G, <NRf> is a positive integer or real number from 0 to 21.00 volts for Output 1 and Output 2 and 0 to 6.50 volts for Output 3.

Returns

<NR3>

Examples

SOURce:VOLTage 3.5 sets the voltage limit to 3.5 volts.

SOURce:VOLTage? returns 0.200E+1 if the voltage limit setting is 2 volts.

[SOURce:]VOLTage:PROTection[:LEVel]

Sets the overvoltage protection level. The query form returns the present setting of the overvoltage protection circuit.

Syntax

```
[SOURce:]VOLTage:PROTection[:LEVel] <NRf>|MAXimum|
MINimum
[SOURce:]VOLTage:PROTection[:LEVel]?
```

Parameters

On the PS2520G, <NRf> ranges from 0 to 38.50 volts for Output 1 and Output 2 and 0 to 7.00 volts for Output 3.

On the PS2521G, <NRf> ranges from 0 to 22.50 volts for Output 1 and Output 2 and 0 to 7.00 volts for Output 3.

Returns

<NR3>

Examples

VOLTage:PROTection 24.5 sets the overvoltage protection to 24.5 volts.

VOLTage? returns 0.3100E+2 if the overvoltage protection setting is 31 volts.

*SRE

Sets the contents of the Service Request Enable Register (SRER). The query form returns the contents of the SRER. Bit 6 of the SRER is always zero. The bits on the SRER correspond to the bits on the SBR. Refer to Table 9 on page 50 for an explanation of each bit in the SBR. Refer to page 52 for additional information about enable registers.

Syntax

```
*SRE <NRf>
*SRE?
```

Parameters

<NRf> is an integer from 0 to 255.

Returns

<NR1>

Examples

*SRE 8 sets bits of the SRER to 0000 1000.

*SRE? returns 2 if the SRER is set to 0000 0010.

STATus:OPERation:CONDition? (Query Only)

Returns the contents of the OPERation register. The PS2520G and PS2521G Programmable Power Supplies, however, do not use the OPERation register to report any conditions.

Reading the OPERation register does not affect its contents.

Syntax

```
STATus:OPERation:CONDition?
```

Returns

<NR1>

Examples

STATus:OPERation:CONDition? returns 0.

STATus:OPERation:ENABle

Sets or returns the contents of the OPERation Enable Register. Even though this is a 16-bit register, only 15 bits (bit 0 through bit 14) are used. Bit 15 is always 0.

Syntax

```
STATus:OPERation:ENABle <NRf>
STATus:OPERation:ENABle?
```

Parameters

<NR1> is an integer from 0 to 32767.

Returns

<NR1>

Examples

STATus:OPERation:ENABle 32767 sets all 15 bits of the register high.

STATus:OPERation:ENABle? returns 0 if all 15 bits of the register are low.

STATus:OPERation[:EVENt]?

Returns and clears the contents of the OPERation register. The response is a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:OPERation[:EVENt]?
```

Returns

<NR1>

Examples

STATus:OPERation:EVENt? returns 0.

STATus:OPERation:INSTrument:CONDition? (Query Only)

Returns the contents of the OPERation INSTrument register. The PS2520G and PS2521G Programmable Power Supplies, however, do not use the OPERation INSTrument register to report any conditions.

Reading the OPERation INSTrument register does not affect its contents.

Syntax

```
STATus:OPERation:INSTrument:CONDition?
```

Returns

<NR1>

Examples

STATus:OPERation:INSTrument:CONDition? returns 0.

STATus:OPERation:INSTrument:ENABle

Sets or returns the contents of the OPERation INSTrument Enable Register. Even though this is a 16-bit register, only 15 bits (bit 0 through bit 14) are used. Bit 15 always reads 0.

Syntax

```
STATus:OPERation:INSTrument:ENABle <NRf>
STATus:OPERation:INSTrument:ENABle?
```

Parameters

<NR1> is an integer from 0 to 32767.

Returns

<NR1>

Examples

STATus:OPERation:INSTrument:ENABle 32767 sets all 15 bits of the register high.

STATus:OPERation:INSTrument:ENABle? returns 0 if all 15 bits of the register are low.

STATus:OPERation:INSTrument[:EVENt]?

Returns and clears the contents of the OPERation INSTrument register. The response is a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:OPERation:INSTrument[:EVENt]?
```

Returns

<NR1>

Examples

STATus:OPERation:INSTrument:EVENt? returns 0.

STATus:OPERation:INSTrument:ISUMmary<n>:CONDition? (Query Only)

Returns the contents of the OPERation INSTrument ISUMmary<n> register. The PS2520G and PS2521G Programmable Power Supplies, however, do not use the OPERation INSTrument ISUMmary<n> register to report any conditions.

Reading the OPERation INSTrument ISUMmary<n> register does not affect its contents.

Syntax

```
STATus:OPERation:INSTrument:ISUMmary<n>:CONDition?
```

Returns

0

Examples

STATus:OPERation:INSTrument:ISUMmary1:CONDition? returns 0.

STATus:OPERation:INSTrument:ISUMmary<n>:ENABle

Sets or returns the contents of the OPERation INSTrument ISUMmary<n> Enable Register. Even though this is a 16-bit register, only 15 bits (bit 0 through bit 14) are used. Bit 15 always reads 0.

Syntax

```
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle <NRf>
STATus:OPERation:INSTrument:ISUMmary<n>:ENABle?
```

Parameters

<NR1> is an integer from 0 to 32767.

Returns

<NR1>

Examples

STATus:OPERation:INSTrument:ISUMmary1:ENABle 32767 sets all 15 bits of the register high.

STATus:OPERation:INSTrument:ISUMmary1:ENABle? returns 0 if all 15 bits of the register are low.

STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?

Returns and clears the contents of the OPERation INSTrument ISUMmary<n> register. The response is a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:OPERation:INSTrument:ISUMmary<n>[:EVENt]?
```

Returns

```
<NR1>
```

Examples

`STATus:OPERation:INSTrument:ISUMmary1:EVENt?` returns 0.

STATus:PRESet (No Query Form)

Sets the OPERation and QUEStionable enable registers to zeros, the INSTrument enable registers to ones, and ISUMmary registers to zeros.

Syntax

```
STATus:PRESet
```

STATus:QUEue[:NEXT]? (Query Only)

Reads the next item from the Error and Event Queue. Refer to the error codes in Table 11 on page 55. This query is identical to the `SYSTem:ERRor?` query.

Syntax

```
STATus:QUEue[:NEXT]?
```

Returns

```
<NR1>,<string>
```

Examples

STATus:QUEue? returns 0, "No error" if there are no errors in the queue.

STATus:QUEStionable:CONDition? (Query Only)

Returns the contents of the QUEStionable condition register. Reading the QUEStionable condition register does not affect its contents. This query returns a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:QUEStionable:CONDition?
```

Returns

<NR1>

STATus:QUEStionable:ENABle

Sets or returns the contents of the enable register for the QUEStionable event register. Even though this is a 16-bit register, only 15 bits (bit 0 through bit 14) are used. Bit 15 always reads 0.

Syntax

```
STATus:QUEStionable:ENABle <NR1>
STATus:QUEStionable:ENABle?
```

Parameters

<NR1> is a positive integer from 0 to 32767.

Returns

<NR1>

Examples

STATus:QUEStionable:ENABle 32767 sets all 15 bits of the register high.

STATus:QUESTionable:ENABle? returns 0 if all 15 bits of the register are low.

STATus:QUEStionable[:EVENt]?

Returns and clears the contents of the QUEStionable event register. Reading the QUEStionable register using the query form resets the register. The query returns a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:QUEStionable[:EVENt]?
```

Returns

```
<NR1>
```

Examples

`STATus:QUEStionable:EVENt?` returns 2 if bit number 1 is the only bit set.

STATus:QUEStionable:INSTrument:CONDition? (Query Only)

Returns the contents of the QUEStionable INSTrument condition register. Reading the QUEStionable INSTrument condition register does not affect its contents. This query returns a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:QUEStionable:INSTrument:CONDition?
```

Returns

```
<NR1>
```

STATus:QUEStionable:INSTrument:ENABle

Sets or returns the contents of the enable register for the QUEStionable INSTrument event register. Even though this is a 16-bit register, only 15 bits (bit 0 through bit 14) are used. Bit 15 always reads 0.

Syntax

```
STATus:QUEStionable:INSTrument:ENABle <NR1>
STATus:QUEStionable:INSTrument:ENABle?
```

Parameters

<NR1> is a positive integer from 0 to 32767.

Returns

<NR1>

Examples

STATus:QUEStionable:INSTrument:ENABle 32767 sets all 15 bits of the register high.

STATus:QUESTionable:INSTrument:ENABle? returns 0 if all 15 bits of the register are low.

STATus:QUEStionable:INSTrument[:EVENt]?

Returns and clears the contents of the QUEStionable INSTrument event register. Reading the QUEStionable INSTrument register using the query form resets the register. The query returns a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:QUEStionable:INSTrument[:EVENt]?
```

Returns

<NR1>

Examples

STATus:QUEStionable:INSTrument:EVENt? returns 2 if bit number 1 is the only bit set.

STATus:QUEStionable:INSTrument:ISUMmary<n> :CONDition? (Query Only)

Returns the contents of the QUEStionable INSTrument ISUMmary<n> condition register. Reading the QUEStionable INSTrument

ISUMmary<n> condition register does not affect its contents. This query returns a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:QUEStionable:INSTrument:ISUMmary<n>:CONDition?
```

Returns

<NR1>

STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle

Sets or returns the contents of the enable register for the QUEStionable INSTrument ISUMmary<n> event register. Even though this is a 16-bit register, only 15 bits (bit 0 through bit 14) are used. Bit 15 always reads 0.

Syntax

```
STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle <NR1>
STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle?
```

Parameters

<NR1> is a positive integer from 0 to 32767.

Returns

<NR1>

Examples

STATus:QUEStionable:INSTrument:ISUMmary<n>:ENABle 32767 sets all 15 bits of the register high.

STATus:QUESTionable:INSTrument:ISUMmary<n>:ENABle? returns 0 if all 15 bits of the register are low.

STATus:QUEStionable:INSTrument:ISUMmary<n>[:EVENt]?

Returns and clears the contents of the QUEStionable INSTrument event register. Reading the QUEStionable INSTrument register using

the query form resets the register. The query returns a decimal value that summarizes the binary values of the set bits.

Syntax

```
STATus:QUEStionable:INSTrument:ISUMmary<n>[:EVENt]?
```

Returns

```
<NR1>
```

Examples

`STATus:QUEStionable:INSTrument:ISUMmary<n>:EVENt?` returns 2 if bit number 1 is the only bit set.

*STB? (Query Only)

Returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. Refer to Figure 10 on page 49.

Syntax

```
*STB?
```

Returns

```
<NR1>
```

Examples

`*STB?` returns 96 if the SBR contains the binary value 0110 0000.

SYSTem:AUTO

Sets or returns automatic sequence setting.

Syntax

```
SYSTem:AUTO <Boolean>
SYSTem:AUTO?
```

Parameters

0|1|OFF|ON
OFF or 0 sets AUTO off, ON or 1 sets AUTO on.

Returns

0|1

Examples

SYSTem:AUTO 1 sets AUTO SEQ on.

SYSTem:ERRor? (Query Only)

Reads the next item from the Error and Event Queue. Refer to the error codes in Table 11 on page 55. This query is identical to the STATus:QUEue[:NEXT]? query.

Syntax

SYSTem:ERRor?

Returns

<NR1>, <string>

Examples

SYSTem:ERRor? returns -300, "Device-specific error; overvoltage protection error" if the overvoltage protection circuit has disabled the outputs.

SYSTem:VERSion? (Query Only)

Returns the SCPI version to which the power supply complies.

Syntax

SYSTem:VERSion?

Returns

1994.0

*TST? (Query Only)

Tests RAM, ROM, DAC, and ADC components.

Syntax

```
*TST?
```

Returns

```
0|-300
```

Examples

`*TST?` returns `0` if the test is successful.

`*TST?` returns `-300` if the test fails.

*WAI (No Query Form)

The IEEE 488.2 standard requires the WAI command as part of the standard command set. WAI prevents the programmable instrument from executing further commands or queries until all pending operations finish. This command, however, does not serve a useful function on the PS2520G and PS2521G Programmable Power Supplies The power supplies do not have any operations that require additional time to finish.

Syntax

```
*WAI
```

Status and Events

The PS2520G and PS2521G Programmable Power Supplies provide a status and event reporting system for the GPIB interface. Various registers and queues make up this system. This section explains how these registers and queues work to inform you of significant events.

System Structure

Figure 7 on page 43 is a simplified diagram of the status and event reporting system. In reality, each component of the diagram represents a set of registers and queues that read, report, or enable the occurrence of certain events within the system.

Status reporting begins when a specific event in the programmable power supply sets a bit in a *status register.* Reading the status registers tells you what types of events have occurred.

Each bit in the status register corresponds to a bit in an *enable register*; the enable bit must be high for the event to be reported to the Status Byte Register.

The Output Queue stores and reports query responses. The Error/Event Queue stores and reports error messages.

A Service Request (SRQ) is the last event to occur. The SRQ requests an interrupt on the GPIB to report events to the system controller.

The following sections explain the registers and queues in greater detail.

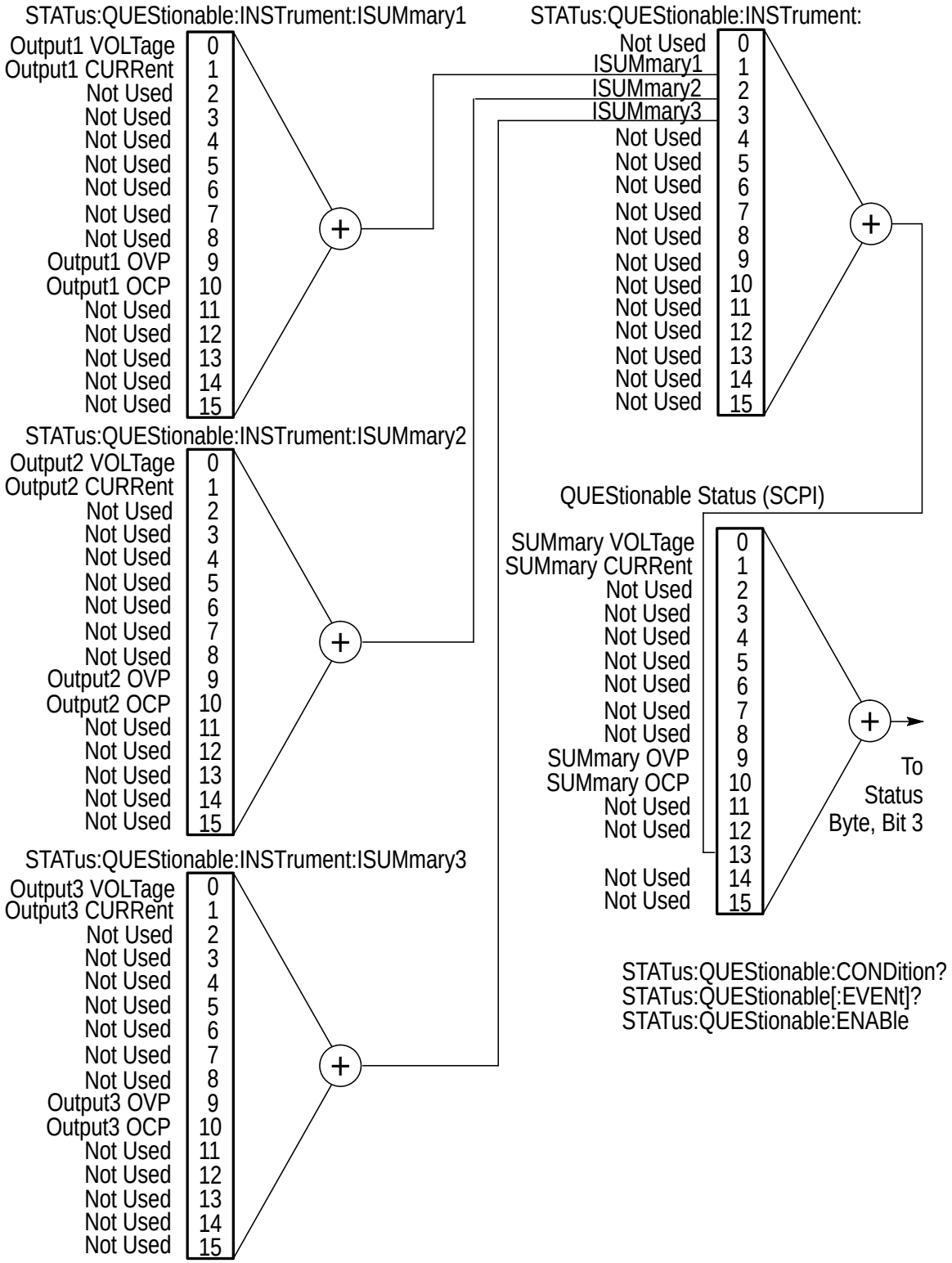

Figure 6: QUEStionable INSTrument Registers

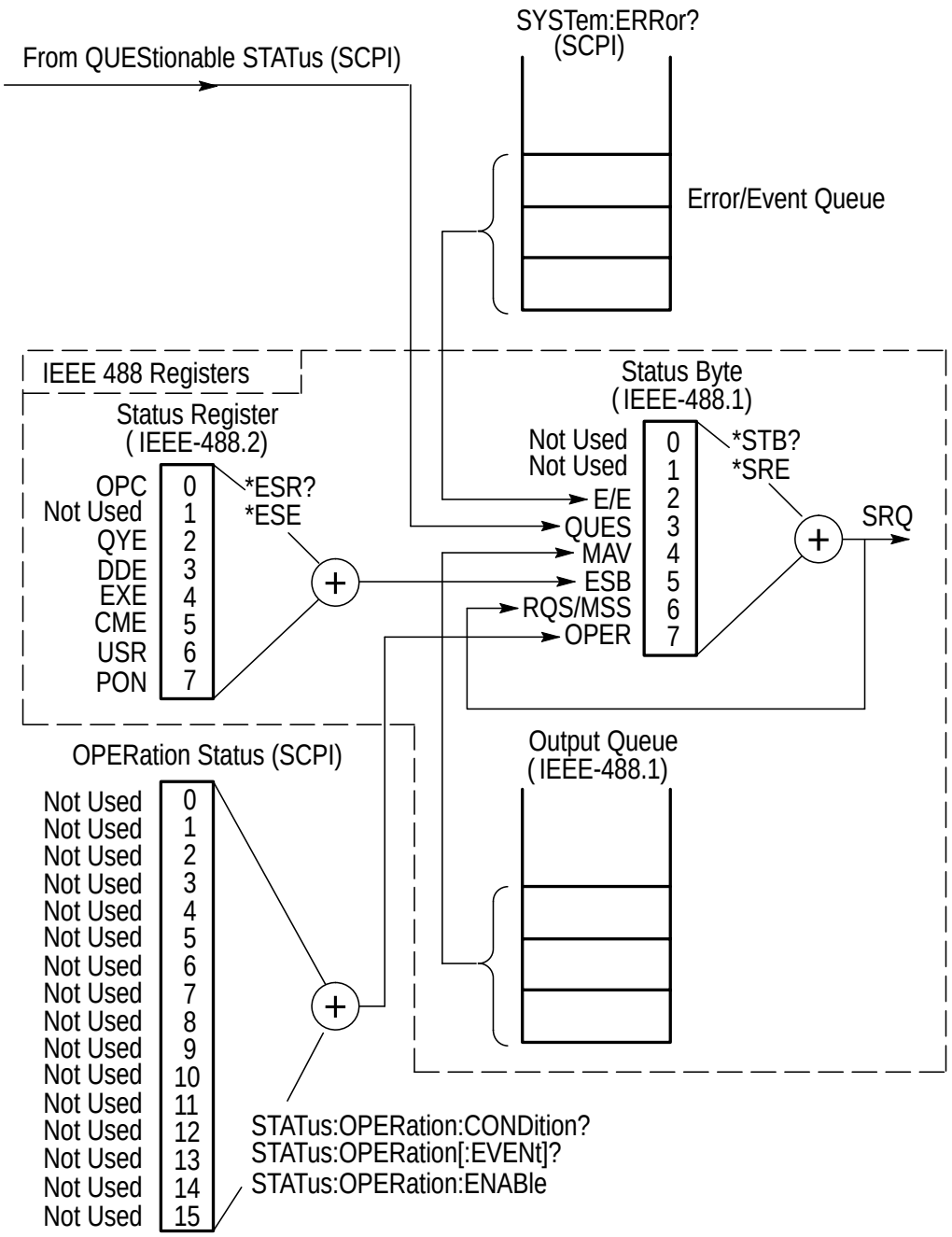

Figure 7: Status and Event System

Status Registers

The programmable power supplies use status registers that are defined by SCPI, IEEE-488.1, and IEEE-488.2 standards.

SCPI Status Registers

The power supplies include two status registers defined by the SCPI standard:

- OPERation Status Registers (CONDition, EVENt, and ENABle)
- QUEStionable Status Registers (CONDition, EVENt, and ENABle)

The STATus subsystem (Figure 8) is the hierarchical set of commands that read the SCPI defined status registers.

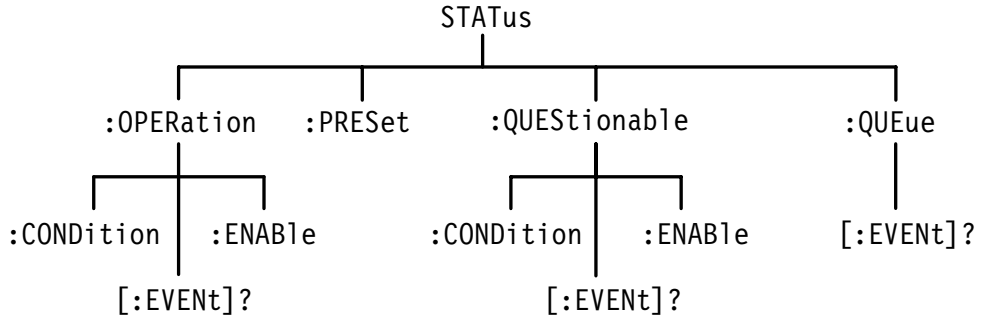

Figure 8: STATus Hierarchy of SCPI Defined Registers

The lower-level nodes :OPERation and :QUEStionable each have three 16-bit registers: CONDition, EVENt, and ENABle. Figure 9 shows the sequential relationship between these three types of registers and the commands that relate to each register.

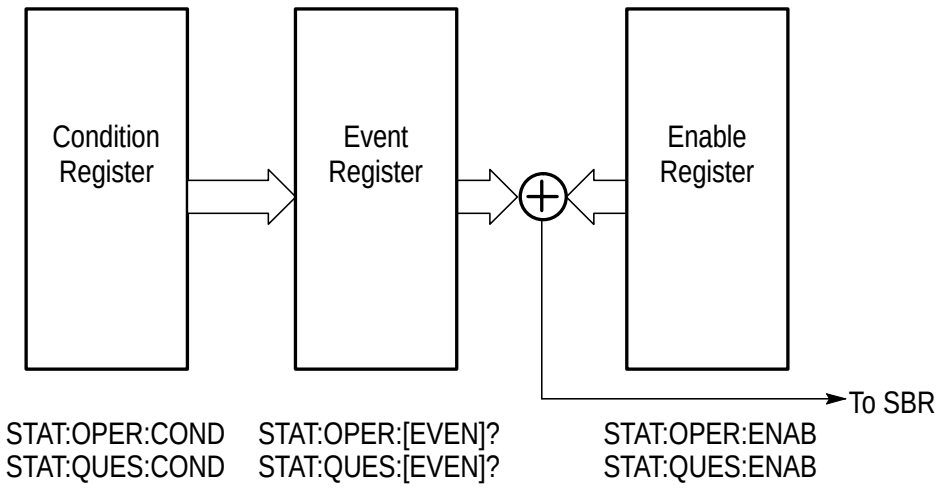

Figure 9: Status Registers and Related Commands

The CONDition register is a read-only register that monitors the current state of the instrument. The CONDition register updates in real time and the inputs are not latched or buffered. When a condition monitored by the CONDition register becomes true, the bit for that condition also becomes true (1). When the condition is false, the bit is 0.

The read-only EVENt register latches any false-to-true change in condition. Once the bit in the EVENt register is set, it is no longer affected by changes in the corresponding bit of the CONDition register. The bit remains set until the controller reads it. The command `*CLS` (Clear Status) clears the EVENt registers.

ENABle registers control the reporting of events latched in the EVENt registers. The QUEStionable Enable Register, for example, sets the summary bit (3) of the Status Byte Register true only if one or more of the bits in the QUEStionable Event Register are true. The programmable power supply can assert SRQ (Service Request) on the GPIB only after the ENABle register sets the summary bit true.

OPERation Status Registers. Although the OPERation Status Registers are present, the programmable power supplies do not use them to report any status information.

QUEStionable Status Register. Table 6 shows the bit designations of the 16 bit QUEStionable Status Register.

Table 6: QUEStionable Status Register

Bit Number	Name	Condition (Bit is True)
0	VOLTage	Constant Current (CC). One or more of Channel 1, Channel 2, or Channel 3 voltages are unregulated.
1	CURRent	Constant Voltage (CV). One or more of Channel 1, Channel 2, or Channel 3 currents are unregulated.
2	—	Not used.
3	—	Not used.
4	—	Not used.
5	—	Not used.
6	—	Not used.
7	—	Not used.
8	—	Not used.
9	OVP	Overvoltage protection tripped.
10	OCP	Overcurrent protection tripped.
11	—	Not used.
12	—	Not used.
13	—	Event on STATus:QUEStionable:INSTrument Register.
14	—	Not used.
15	—	Not used.

The command `STATus:QUEStionable:CONDition?` reads the QUEStionable CONDition register but does not clear it.

The command `STATus:QUEStionable[:EVENt]?` reads and clears the QUEStionable EVENt Status Register.

QUEStionable INSTrument Status Register. Table 7 shows the bit designations of the 16-bit QUEStionable INStrument Status Register.

Table 7: QUEStionable INSTrument Status Register

Bit Number	Name	Condition (Bit is True)
0	—	Not used.
1	ISUMmary1	Summary of Channel 1 events.
2	ISUMmary2	Summary of Channel 2 events.
3	ISUMmary3	Summary of Channel 3 events.
4	—	Not used.
5	—	Not used.
6	—	Not used.
7	—	Not used.
8	—	Not used.
9	—	Not used.
10	—	Not used.
11	—	Not used.
12	—	Not used.
13	—	Not used.
14	—	Not used.
15	—	Not used.

The command `STATus:QUEStionable:INSTrument:CONDition?` reads the QUEStionable INSTrument CONDition register but does not clear it.

The command `STATus:QUEStionable:INSTrument[:EVENt]?` reads and clears the QUEStionable INSTrument EVENt Status Register.

QUEStionable INSTrument ISUMmary<n> Status Register. Table 8 shows the bit designations of the three16-bit QUEStionable INSTrument ISUMmary<n> Status Registers (ISUMmary1, ISUMmary2, and ISUMmary3).

Table 8: QUEStionable INSTrument ISUMmary<n> Status Register

Bit Number	Name	Condition (Bit is True)
0	Output<n> Voltage	Output<n> Constant Current (CC), voltage unregulated.
1	Output<n> Current	Output<n> Constant Voltage (CV), current unregulated.
2	—	Not used.
3	—	Not used.
4	—	Not used.
5	—	Not used.
6	—	Not used.
7	—	Not used.
8	—	Not used.
9	Output<n> OVP	Output<n> overvoltage protection tripped.
10	Output<n> OCP	Output<n> overcurrent protection tripped.
11	—	Not used.
12	—	Not used.
13	—	Not used.
14	—	Not used.
15	—	Not used.

The command `STATus:QUEStionable:INSTrument:ISUMmary<n>:CONDition?` reads the QUEStionable INSTrument ISUMmary<n> CONDition register but does not clear it.

The command `STATus:QUEStionable:INSTrument:ISUMmary<n>[:EVENt]?` reads and clears the QUEStionable INSTrument ISUMmary<n> EVENt Status Register.

IEEE-488.1 and IEEE-488.2 Status Registers

The power supplies also include two status registers defined by IEEE-488.1 and IEEE-488.2 standards:

- Status Byte Register (SBR)
- Standard Event Status Register (SESR)

Status Byte Register (SBR). The SBR (Figure 10) summarizes the status of all other registers and queues.

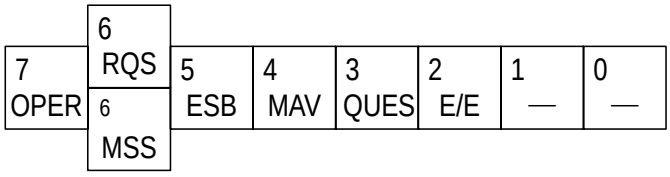

Figure 10: Status Byte Register (SBR)

Use a serial poll or the *STB? query to read the contents of the SBR. The bits in the SBR are set and cleared depending on the contents of the Standard Event Status Register (SESR), the Event Status Enable Register (ESER), and the Output Queue. When you use a serial poll to obtain the SBR, bit 6 is the MSS bit. Reading the SBR does not clear the bits.

Table 9 explains the function of each bit in the SBR.

Table 9: SBR Bit Functions

Bit	Function
0 (LSB)	Not used. This bit is always zero.
1	Not used. This bit is always zero.
2	E/E (Error and Event) indicates an error code is waiting to be read in the Error Event Queue
3	QUES (QUEStionable) is the summary bit for the QESR (QUEStionable Event Status Register). When this bit is high it indicates that status is enabled and present in the QESR.
4	MAV (Message Available) indicates that output is available in the output queue.
5	ESB (Event Status Bit) is the summary bit for the Standard Event Status Register (SESR). When this bit is high it indicates that status is enabled and present in the SESR.
6	RQS (Request Service) is obtained from a serial poll. This bit shows that the power supply requests service from the GPIB controller. MSS (Master Status Summary) is obtained from *STB? query. This bit indicates another bit in the SBR has been enabled and is set.
7 (MSB)	OPER (OPERation) is the summary bit for the OESR (OPERation EVENt Status Register). This bit is never set in the PS2520G and PS2521G Programmable Power Supplies.

Standard Event Status Register (SESR). Figure 11 shows the SESR. The SESR records five types of events that can occur within the power supply.

7	6	5	4	3	2	1	0
PON	USR	CME	EXE	DDE	QYE	—	OPC

Figure 11: The Standard Event Status Register (SESR)

Use the *ESR? query to read the SESR. Reading the SESR clears the bits of the register so that the register can accumulate information about new events.

Table 10 explains the function of each bit in the SESR.

Table 10: SESR Bit Functions

Bit	Function
0 (LSB)	OPC (Operation Complete) shows that the operation is complete. This bit is set when all pending operations are completed following an *OPC command.
1	Not used. This bit is always zero.
2	QYE (Query Error) indicates a command or query protocol error. Error messages are listed in Table 11 (“–4XX” errors).
3	DDE (Device Error) shows that a device error occurred. Error messages are listed in Table 11 (“–3XX” errors).
4	EXE (Execution Error) shows that an error occurred while the power supply was executing a command or query. Table 11 lists the error messages (“–2XX” errors).
5	CME (Command Error) shows that an error occurred while the power supply was parsing a command or query. Table 11 on page 55 lists the error messages (“–1XX” errors).
6	USR (User Request) indicates the LOCAL button was pushed.
7 (MSB)	PON (Power On) shows that the power supply was powered on.

Enable Registers

The enable registers determine whether certain events are reported to the Status Byte Register and SRQ. The programmable power supply has the following enable registers:

- Event Status Enable Register (ESER)
- Service Request Enable Register (SRER)
- OPERation Enable Register
- QUEStionable Enable Register

The enable registers perform a logical OR function; when one of the bits of the enable registers is high and the corresponding bit in the status register is high, the output that controls the set bit of the Status Byte Register is high.

Various commands set the bits in the enable registers. The following sections describe the enable registers and the commands that set them.

Event Status Enable Register (ESER)

The ESER (Figure 12) controls which types of events are summarized by the Event Status Bit (ESB) in the SBR. The bits of the ESER correspond to the bits of the SESR. Refer to Table 10 on page 51 for an explanation of each bit in the SESR.

Use the *ESE command to set the bits in the ESER. Use the *ESE? query to read it.

7	6	5	4	3	2	1	0
PON	USR	CME	EXE	DDE	QYE	—	OPC

Figure 12: Event Status Enable Register (ESER)

Service Request Enable Register (SRER)

The SRER (Figure 13) controls which bits in the SBR generate a service request.

Use the *SRE command to set the SRER. Use the *SRE? query to read it.

The bits of the SRER correspond to the bits of the SBR. Refer to Table 9 on page 50 for an explanation of each bit in the SBR.

7	6	5	4	3	2	1	0
OPER	—	ESB	MAV	QUES	E/E	—	—

Figure 13: Service Request Enable Register (SRER)

OPERation Enable Register

Even though the OPERation Enable Register is present in the programmable power supplies, the OPERation registers do not report any conditions.

QUEStionable Enable Register

The QUEStionable Enable Register controls which types of events are summarized by the QUES status bit in the SBR. Use the STATus:QUEStionable:ENABle command to set the bits in the QUEStionable Enable register. Use the STATus:QUEStionable:ENABle? query to read it. Refer to Table 6 on page 46 for a description of each of the bits in this register.

Queues

The programmable power supplies contain two queues: the Output Queue and the Error/Event queue.

Output Queue

Following IEEE 488.2 protocols, the programmable power supplies store query responses in the Output Queue. The power supply clears and resets this queue each time it receives a new command or query message after a message terminator. The controller must read a query response before it sends the next command (or query) or it loses responses to earlier queries.

Error/Event Queue

When an error or event occurs, the Error/Event Queue stores the message and sets bit 2 of the Status Byte Register high. Enabling this bit by using the `*SRE 4` command causes the event to signal the GPIB controller with a Service Request (SRQ) signal.

The Error/Event Queue stores and reports the messages on a first-in-first-out basis. The `SYSTem:ERRor?` or the `STATus:QUEue[:NEXT]?` query reads the next item from the Error/Event Queue. If the Error/Event Queue overflows, the last message is `-350, "Queue overflow"`; the queue cannot store or report subsequent messages until it is read or cleared.

Error Messages

Table 11 lists the SCPI error messages for the programmable power supplies. The listing includes the equivalent front-panel error code along with a description of the error message.

Table 11: Error Messages

SCPI Error Code and Description	Front Panel Error Code	SESR Bit
0, “No error”	—	—
–100, “Command Error”	—	5
–108, “Parameter not allowed”	—	5
–109, “Missing parameter”	—	5
–121, “Invalid character in number”	—	5
–124, “Too many digits”	—	5
–200, “Execution error”	—	4
–200, “Execution error; STEP error”	—	4
–221, “Setting conflict; Timer setting error”	–064	4
–221, “Setting conflict; Overvoltage protection setting error”	–065	4
–221, “Setting conflict; Address setting error”	–066	4
–221, “Setting conflict; Voltage setting error”	–067	4
–221, “Setting conflict; Current setting error”	–068	4
–221, “Setting conflict; Recall setting error”	–069	4
–221, “Setting conflict; Store setting error”	–070	4
–221, “Setting conflict; STEP voltage or current setting error”	—	4
–222, “Data out of range; Voltage too large”	–016	4
–222, “Data out of range; Current too large”	–017	4
–222, “Data out of range; Voltage too small”	–018	4
–222, “Data out of range; Current too small”	–019	4
–222, “Data out of range”	—	4
–240, “Hardware error”	—	4
–300, “Device-specific error; Overcurrent protection error”	–012	3
–300, “Device-specific error; Overvoltage protection error”	–013	3

Table 11: Error Messages (Cont.)

SCPI Error Code and Description	Front Panel Error Code	SESR Bit
–300, “Device-specific error; Calibration current full-scale error”	–091	3
–300, “Device-specific error; Calibration voltage full-scale error”	–092	3
–300, “Device-specific error; Calibration overvoltage protection full-scale error”	–093	3
–300, “Device-specific error; Calibration overvoltage protection offset error”	–094	3
–310, “System error”	—	3
–313, “Calibration memory lost”	—	3
–330, “Self-test failed”	—	3
–330, “Self-test failed; CPU test error”	–001	3
–330, “Self-test failed; RAM test error”	–002	3
–330, “Self-test failed; ROM test error”	–003	3
–330, “Self-test failed; DAC/ADC test error”	–005	3
–350, “Queue overflow”	—	—
–410, “Query INTERRUPTED”	—	2
–420, “Query UNTERMINATED”	—	2
–430, “Query DEADLOCKED”	—	2
–440, “Query UNTERMINATED after indefinite response”	—	2

Index

M

N

O

P

Q

R

S

T

U

W