PWS4205, PWS4305, PWS4323, PWS4602, and PWS4721 Linear DC Power Supplies

Programmer Manual

077-0481-03

Tektronix

PWS4205, PWS4305, PWS4323, PWS4602, and PWS4721 Linear DC Power Supplies

Programmer Manual

www.tektronix.com

077-0481-03

Tektronix

Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

Contacting Tektronix

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Table of Contents

Preface

This programmer manual provides commands, and explains the use of those commands, for remotely controlling the PWS4205, PWS4305, PWS4323, PWS4602, and the PWS4721 Linear DC Power Supplies. With this information, you can write computer programs to perform functions, such as setting the controls, taking measurements, performing statistical calculations, and exporting data for use in other programs.

Getting Started

Getting Started

Your power supply has a USB 2.0 high-speed device port to control the power supply using the USBTMC protocol. The USBTMC protocol allows USB devices to communicate using IEEE-488.2 style messages.

You can also remotely communicate between your power supply and PC over GPIB using the TEK-USB-488 Adapter.

Using the USB

Start by connecting an appropriate USB cable between the USB 2.0 high-speed device port on the rear panel of your power supply and a PC.

In order for the PC to recognize the power supply, a USBTMC driver must be installed on the PC. A USBTMC driver can be installed on your PC by installing a virtual instrument communications API like TekVISA or NIVISA. These VISAs are available for download from the Tektronix or National Instruments Web sites. Once the USBTMC driver is loaded, your PC will establish communication with the power supply upon USB cable connection.

For further remote control and/or programming use, other software applications may be needed in addition to a VISA and the USBTMC driver.

Using the GPIB

Start by connecting an appropriate USB cable between the USB 2.0 high-speed device port on the rear panel of your power supply and the TEK-USB-488 Adapter host port. Then connect a GPIB cable from the TEK-USB-488 Adapter to your PC.

Supply power to the adapter in one of the following ways:

- Use an optional 5 V DC power adapter connected to the 5 V DC power input on the adapter.
- Use an appropriate USB cable connected to a powered USB host port on your PC and the device port on the TEK-USB-488 Adapter.

GPIB Requirements

Before setting up the power supply for remote communication using the electronic (physical) GPIB interface, you should familiarize yourself with the following recommendations:

- A unique device address should be assigned to each device on the bus. No two devices should share the same device address.
- No more than 15 devices can be connected to any one bus.
- Only one device should be connected for every 6 feet (2 meters) of cable used.
- No more than 65 feet (20 meters) of cable should be used to connect devices to a bus.
- At least two-thirds of the devices on the network should be on while using the network.
- Connect the devices on the network in a star or linear configuration. Do not use loop or parallel configurations.

To Change GPIB Address Settings

Your power supply must have a unique device address to function properly. The default setting for the GPIB configuration is GPIB Address 1. If there is more than one GPIB instrument on the bus, you will need to change the default setting on the power supply. To change the GPIB address settings, do the following:

1. On the instrument front-panel, push the **Shift** button and then the **1** button to access the menu.
2. Press the down arrow key until you see **System** and then press the **Enter** button.
3. Press the down arrow key until you see **Address** and then press the **Enter** button.
4. You can now change the address of your GPIB port. This will set the GPIB address on an attached TEK-USB-488 Adapter.

The power supply is now set up for bidirectional communication with your controller.

Command Timing

The average time it takes to both send and receive every command is approximately 20 ms. In the case of more complex commands, more time may be required to complete transmission.

Command Syntax

You can control the power supply through the USB interface using commands and queries.

This section describes the syntax these commands and queries use and the conventions the power supply uses to process them. The commands and queries themselves are listed by group and alphabetically. (See page 2-7, *Command Groups*.)

You transmit commands to the power supply using the enhanced American Standard Code for Information Interchange (ASCII) character encoding. *Appendix A* contains a chart of the ASCII character set.

The Backus Naur Form (BNF) notation is used in this manual to describe commands and queries. (See Table 2-1.)

Table 2-1: BNF notation

Symbol	Meaning
< >	Defined element
::=	Is defined as
|	Exclusive OR
{ }	Group; one element is required
[ ]	Optional; can be omitted
. . .	Previous element(s) may be repeated
( )	Comment

Command and Query Structure

Commands consist of set commands and query commands (usually simply called commands and queries). Commands change power supply settings or perform a specific action. Queries cause the power supply to return data and information about its status.

Most commands have both a set form and a query form. The query form of the command is the same as the set form except that it ends with a question mark. For example, the set command STATus:OPERation:ENAble has a query form STATus:OPERation:ENAble?. Not all commands have both a set and a query form; some commands are set only and some are query only.

A command message is a command or query name, followed by any information the power supply needs to execute the command or query. Command messages consist of five different element types. (See Table 2-3.)

Table 2-2: Command message elements

Symbol	Meaning
<Header>	The basic command name. If the header ends with a question mark, the command is a query. The header may begin with a colon (:) character; if the command is concatenated with other commands the beginning colon is required. The beginning colon can never be used with command headers beginning with a star (*).
<Mnemonic>	A header subfunction. Some command headers have only one mnemonic. If a command header has multiple mnemonics, they are always separated from each other by a colon (:) character.
<Argument>	A quantity, quality, restriction, or limit associated with the header. Not all commands have an argument, while other commands have multiple arguments. Arguments are separated from the header by a <Space>. Arguments are separated from each other by a <Comma>.
<Comma>	A single comma between arguments of multiple-argument commands. It may optionally have white space characters before and after the comma.
<Space>	A white space character between command header and argument. It may optionally consist of multiple white space characters.

The following figure shows the five command message elements.

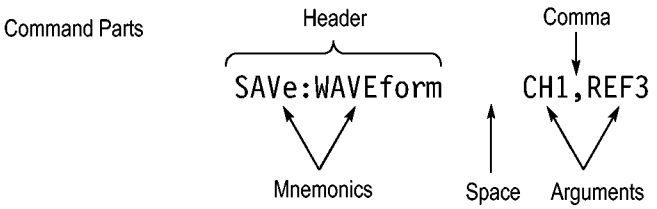


Figure 2-1: Command message elements

Commands

Commands cause the power supply to perform a specific function or change one of its settings. Commands have the structure:

[:]<Header>[<Space><Argument>[<Comma><Argument>]...]

A command header is made up of one or more mnemonics arranged in a hierarchical or tree structure. The first mnemonic is the base or root of the tree and each subsequent mnemonic is a level or branch off of the previous one. Commands at a higher level in the tree may affect those at a lower level. The leading colon (:) always returns you to the base of the command tree.

Queries

Queries cause the power supply to return information about its status or settings. Queries have the structure:

[:]<Header>

[:]<Header>[<Space><Argument>[<Comma><Argument>]...]

You can specify a query command at any level within the command tree unless otherwise noted. These branch queries return information about all the mnemonics below the specified branch or level.

Query Responses

When a query is sent to the power supply, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format, as shown in the following table.

Table 2-3: Query response examples

Query	Response
MEASure:VOLTage:DC?	5.0011
SOURce:FUNCtion:MODE?	LIST

Command Entry

Follow these general rules when entering commands:

- Enter commands in upper or lower case.
- You can precede any command with white space characters. White space characters include any combination of the ASCII control characters 00 through 09 and 0B through 20 hexadecimal (0 through 9 and 11 through 32 decimal).
- The power supply ignores commands that consists of just a combination of white space characters and line feeds.

SCPI Commands and Queries

The power supply uses a command language based on the SCPI standard. The SCPI (Standard Commands for Programmable Instruments) standard was created by a consortium to provide guidelines for remote programming of instruments. These guidelines provide a consistent programming environment for instrument control and data transfer. This environment uses defined programming messages, instrument responses and data formats that operate across all SCPI instruments, regardless of manufacturer.

The SCPI language is based on a hierarchical or tree structure that represents a subsystem. The top level of the tree is the root node; it is followed by one or more lower-level nodes. (See Figure 2-2.)

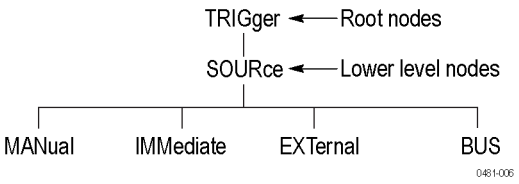


Figure 2-2: Example of SCPI subsystem hierarchy tree

You can create commands and queries from these subsystem hierarchy trees. Commands specify actions for the instrument to perform. Queries return measurement data and information about parameter settings.

Message Terminators

This manual uses the term <EOM> (End of message) to represent a message terminator.

USB End of Message (EOM) terminators. See the USB Test and Measurement Class Specification (USBTMC) section 3.2.1 for details. The power supply terminates messages by setting the EOM bit in the USB header of the last transfer of a message to the host (USBTMC Specification section 3.3.1), and by terminating messages with a LF.

When receiving, the power supply expects a LF and an asserted EOM bit as a message terminator.

Parameter Types

Many power supply commands require parameters. Parameters are indicated by angle brackets, such as <file_name>. There are several different types of parameters, as listed in the following table. The parameter type is listed after the parameter. Some parameter types are defined specifically for the arbitrary/function generator command set and some are defined by SCPI. (See Table 2-4.)

Table 2-4: Types of parameters

Parameter type	Description	Example
boolean	Boolean numbers or values	ON or ≠ 0 OFF or 0
discrete	A list of specific values	MIN, MAX
NR1 numeric	Integers	0, 1, 15, -1
NR2 numeric	Decimal numbers	1.2, 3.141516, -6.5
NR3 numeric	Floating point numbers	3.1415E-9, -16.1E5
NRf numeric	Flexible decimal number that may be type NR1, NR2, or NR3	See NR1, NR2, NR3 examples in this table
string	Alphanumeric characters (must be within quotation marks)	“Testing 1, 2, 3”

Abbreviating Commands, Queries, and Parameters

You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these commands as a combination of upper and lower case letters. The upper case letters indicate the accepted short form of a command, as shown in the following figure. The accepted short form and the long form are equivalent and request the same action of the instrument.

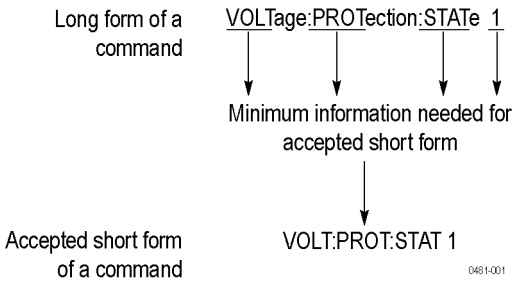


Figure 2-3: Example of abbreviating a command

Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, then add a semicolon (;), and finally add more commands or queries and semicolons until you are done. If the command following a semicolon is a root node, precede it with a colon (:). The following figure illustrates a chained message consisting of several commands and queries. The chained message should end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.

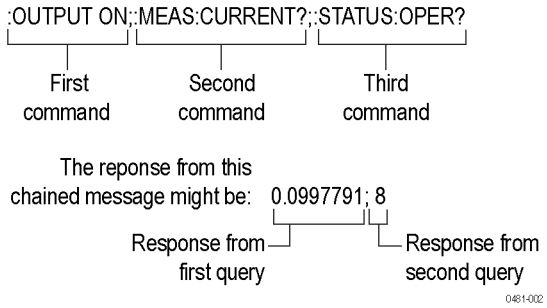


Figure 2-4: Example of chaining commands and queries

If a command or query has the same root and lower-level nodes as the previous command or query, you can omit these nodes. In the following figure, the second command has the same root node (STAT:QUES) as the first command, so these nodes can be omitted.

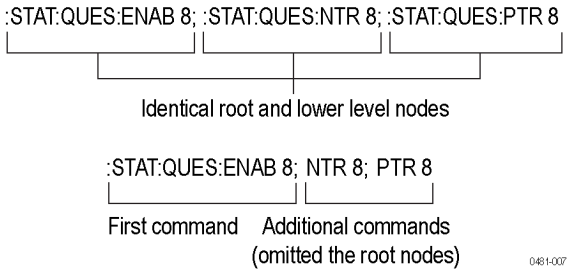


Figure 2-5: Example of omitting root and lower level nodes

General Rules for Using SCPI Commands

The following are three general rules for using SCPI commands, queries, and parameters:

- You can use single (‘ ’) or double (“ ”) quotation marks for quoted strings, but you cannot use both types of quotation marks for the same string.

correct	“This string uses quotation marks correctly.”
correct	‘This string also uses quotation marks correctly.’
incorrect	“This string does not use quotation marks correctly.’

- You can use upper case, lower case, or a mixture of both cases for all commands, queries, and parameters.

  `:SOURCE:FREQUENCY 10MHZ`
  is the same as
  `:source:frequency 10mhz`
  and
  `:SOURCE:frequency 10MHZ`

NOTE. *Quoted strings are case sensitive.*

- No embedded spaces are allowed between or within nodes.

correct	`:OUTPUT:FILTER:LPASS:FREQUENCY 200MHZ`
incorrect	`:OUTPUT: FILTER: LPASS:FREQUENCY 200MHZ`

Command Groups

This manual lists the power supply commands in two ways. First, it presents them by functional groups. Then, it lists them alphabetically. The functional group list starts below. The alphabetical list provides detail on each command. (See page 2-57, Commands Listed in Alphabetical Order.)

The power supply interface conforms to Tektronix standard codes and formats except where noted. The GPIB interface also conforms to IEEE Std 488.2–1987 except where noted. The USB interface also conforms to USB Test and Measurement Class, Subclass USB488 Specification, except where noted. Arguments are not mentioned in the group command descriptions, but are listed under the commands in the *Commands Listed in Alphabetical Order* section of this manual. (See page 2-13.)

Status Commands

Status commands let you determine the status of the power supply and control events.

Several commands and queries are common to all devices on the GPIB or USB bus. These commands and queries are defined by IEEE Std. 488.2-1987 and Tek Standard Codes and Formats 1989, and begin with an asterisk (*) character.

Table 2-5: Status commands

Command	Description
*CLS	Clear all event registers and queues
*ESR?	Return standard event status register
*ESE	Set/query standard event status enable register
*IDN?	Return identification information in IEEE 488.2 notation
*RST	Resets to known settings, but does not purge stored settings
*PSC	Set/query power-on status clear
*SRE	Set/query service request enable register
*STB?	Read status byte
STATus:QUEStionable:CONDition?	Return questionable condition register. When a bit of the quest condition changes, the corresponding bit value in the quest event register will be set to 1.

Table 2-5: Status commands (cont.)

Command	Description
STATus:QUEStionable:ENABle	Set/query questionable enable register. This parameter determines which bit of the quest event register is set to 1. If a QUES condition changes, the QUES bit of status byte register will be set to 1.
STATus:QUEStionable[:EVENt]?	Return questionable event register
STATus:QUEStionable:PTRansition	Edit the rising edge parameter of quest event register
STATus:QUEStionable:NTRansition	Edit the trailing edge parameter of quest event register
STATus:OPERation:CONDition?	Return operation condition register. When a parameter of the operation condition register changes, the corresponding bit in the operation event register will be set to 1.
STATus:OPERation:ENABle	Set/query operation enable register. The parameter determines which bit value of quest event register is set to 1. If a OPER condition changes, the OPER bit of the status byte register will be set to 1.
STATus:OPERation[:EVENt]?	Return operation event register

Save and Recall Commands

Save and recall commands allow you to save the active settings to one of the settings memories within the power supply, and recall those settings at a later time.

Table 2-6: Save and recall commands

Header	Description
*SAV	Save instrument setting to setup memory
*RCL	Recall instrument setting from setup memory

System Commands

Table 2-7: System commands

Header	Description
SYSTem:POSetup	Set power-on parameters
SYSTem:VERSion?	Return version information
SYSTem:ERRor?	Return error code and error information
SYSTem:KEY	Set key operation

Table 2-7: System commands (cont.)

Header	Description
SYSTem:REMote	Set to remote mode
SYSTem:RWLock	Set to remote mode and lock front-panel
SYSTem:LOCal	Set to front-panel control mode
CONFigure:SOUNd[:STATe]	Set key beep sound on or off

Diagnostic Commands

The power supply includes a self test function that may be used to confirm that it is functioning as expected. A table of error codes that may be returned by the self test are given in the *Messages and Codes* section. (See page 3-8.)

Table 2-8: Diagnostic commands

Header	Description
*TST?	Perform self-test and return result status

Synchronization Commands

Table 2-9: Synchronization commands

Header	Description
*OPC	Set/query operation complete
*WAI	Wait to continue

Trigger Commands

Trigger commands are used to determine the timing of list mode sequences.

Table 2-10: Trigger commands

Header	Description
TRIGger[:IMMediate]	Forces an immediate trigger event
TRIGger:SOURce	Set/query the source of trigger signal
*TRG	Generates a trigger event

Measurement Commands

Measurement commands are used to query parameters. The MEASure commands initiate and execute a complete measurement cycle and are recommended for measuring voltage and current at the outputs of the power supply. FETCh commands do not initiate a new measurement cycle but rely on measurements stored in the communication buffers of the power supply. The FETCh commands are provided for voltage and current measurements to maintain compatibility with other instruments. Output power, however, is only available using a FETCh command.

Table 2-11: Measurement commands

Command	Description
MEASure:VOLTage[:DC]?	Query the measured output voltage
MEASure:CURRent[:DC]?	Query the measured output current
FETCh:CURRent[:DC]?	Query the measured output current
FETCh[:SCALar]:POWer?	Query the measured output power
FETCh:VOLTage[:DC]?	Query the measured output voltage

Source Commands

These commands allow you to set various output parameters. Some of the commands are used to configure protection functions like OVP and Max Voltage.

Table 2-12: Source commands

Command	Description
[SOURce:]CURRent[:LEVel]	Set the current value in units of A or mA
[SOURce:]OUTPut:TIMer[:STATe]	Set the state of the output timer
[SOURce:]OUTPut[:STATe]	Set power supply output on or off
[SOURce:]OUTPut:TIMer:DELay	Set the time of output timer
[SOURce:]OUTPut:PON[:STATe]	Set output on or off at power on
[SOURce:]VOLTage:RANGe	Set maximum allowable voltage level
[SOURce:]VOLTage[:LEVel]	Set voltage value
[SOURce:]VOLTage:PROTection:STATe	Activates overvoltage protection (OVP)
[SOURce:]VOLTage:PROTection[:LEVel]	Set overvoltage protection (OVP) threshold
[SOURce:]OUTPut:PROTection:CLEar	Clear a trip condition caused by overvoltage or over-temperature, or reset a latched remote-inhibit state

List Commands

List commands configure automatic test sequences for execution on the power supply. Programming examples for the [SOURce:]LIST:STEP command are provided in the appendix. (See page B-1, *Programming Examples*.)

Table 2-13: List commands

Command	Description
[SOURce:]LIST:COUNt	Set the number of times a list will execute (loop)
[SOURce:]LIST:CURRent[:LEVel]	Set the current for a list step
[SOURce:]LIST:MODE	Set the trigger condition for executing the list file
[SOURce:]LIST:RCL	Recall the list file saved before from the register
[SOURce:]LIST:SAVe	Save the list file into register
[SOURce:]LIST:STEP	Set the number of steps for the list operation
[SOURce:]LIST:WIDth	Set the step time
[SOURce:]LIST:VOLTage[:LEVel]	Set the voltage for a list step
[SOURce:]FUNCtion:MODE	Configure for command fixed mode or list mode

NOTE. *For details on setting values for each list command, refer to the alphabetical listing of commands.*

Digital I/O Commands

Digital I/O commands allow you to configure, set, and query the I/O pins on the rear panel digital input and output lines.

Table 2-14: Digital I/O commands

Header	Description
[SOURce:]DIGital:FUNCtion	Set rear panel port function
[SOURce:]DIGital:DATA	Set/query rear panel port output state
[SOURce:]OUTPut:DFI:SOURce	Set output source of DFI
[SOURce:]OUTPut:RI:MODE	Set input mode of the remote inhibit (RI) input pin

Bus Command Group

These commands are used in systems that include a TEK-USB-488 bus adapter module. The TEK-USB-488 adapter module allows a GPIB controller to communicate with the power supply.

Table 2-15: Bus commands

Header	Description
GPIBUsb:ADDress?	Allows TEK-USB-488 to query the GPIB Primary Address set for this instrument
GPIBUsb:ID?	Query the ID string of the TEK-USB-488 bus adapter module

Commands Listed in Alphabetical Order

You can use commands to either set instrument features or query instrument values. You can use some commands to do both, some only to set and some only to query. This document marks set-only commands with the words “No Query Form” included with the command name. It marks query-only commands with a question mark appended to the header, and includes the words “Query Only” in the command name.

This document spells out headers, mnemonics, and arguments with the minimal spelling shown in uppercase. For example, to use the abbreviated form of the `MEASure:SCALar:VOLTage:DC?` command, type `MEAS:SCAL:VOLT:DC?`.

*CLS (No Query Form)

The *CLS command clears all event registers and queues.

Group Status

Syntax `*CLS`

Related Commands *ESR?, *STB?

CONFigure:SOUNd[:STATe]

This command turns the key beep sound on or off.

Group System

Syntax

```
CONFigure:SOUNd[:STATe] {0|1|ON|OFF}
CONFigure:SOUNd[:STATe]?
```

Arguments `0` or `OFF` turns the key beep sound off.

`1` or `ON` turns the key beep sound on.

Returns `0|1`

Examples

```
CONF:SOUN:OFF
```

`CONF:SOUN?` might return 0, indicating the key beeper is turned off.

*ESE

Sets and queries the bits in the Event Status Enable Register (ESER). The ESER is an eight-bit mask register that determines which bits in the Standard Event Status Register(SESR) will set the ESB bit in the Status Byte Register (SBR). (See page 3-1, *Status and Events*.)

Group Status

Syntax

```
*ESE <mask>
*ESE?
```

Related Commands *CLS, *ESR?

Arguments `<mask>::=<NR1>` where:

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

The power-on default for ESER is 0 if *PSC is 1. If *PSC is 0, the ESER maintains its value through a power cycle.

Examples

`*ESE 145` sets the ESER to binary 10010001, which enables the PON, EXE, and OPC bits.

`*ESE` might return the string `*ESE 186`, showing that the ESER contains the binary value 10111010.

*ESR? (Query Only)

Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). (See page 3-1, *Status and Events*.)

Group

Status

Syntax

`*ESR?`

Related Commands

*CLS, *OPC, *SRE,

Returns

`<NR1>`, which is a decimal representation of the contents of the Standard Event Status Register (SESR).

Examples

`*ESR?` might return the value `149`, showing that the SESR contains binary 10010101.

FETCh:CURRent[:DC]? (Query Only)

This command returns the last measured output current stored in the communications buffer of the power supply. A new measurement is not initiated by this command.

Group

Measurement

Syntax

`FETCh:CURRent[:DC]?`

Related Commands

MEASure:CURRent[:DC]?

Returns

`<NR2>`, which gives is the measured output current in amperes.

Examples FETC:CURR? might return 0.09998, which would be the current measured at the output of t he power supply in amperes.

FETCh:VOLTage[:DC]? (Query Only)

This command returns the last measured output voltage stored in the communications buffer of the power supply. A new measurement is not initiated by this command.

Group Measurement

Syntax FETCh:VOLTage[:DC]?

Related Commands MEASure:VOLTage[:DC]?

Returns <NR2> is the measured output voltage in volts.

Examples FETC:VOLT? might return 5.0011, which would be the measured voltage across the power supply outputs in volts.

FETCh[:SCALar]:POWer? (Query Only)

This command returns the last measured output current stored in the communications buffer of the power supply. A new measurement is not initiated by this command. The power calculation in the instrument is performed approximately every 100 ms. Insure that the voltage and current are stable longer than this for good results.

Group Measurement

Syntax FETCh[:SCALar]:POWer?

Returns <NR2> is the measured output power in watts.

Examples FETCh:POW? might return 6.01667, which would be the power measured at the output of t he power supply in watts.

GPIBUsb:ADDress? (Query Only)

For use in systems that include a TEK-USB-488 bus adapter module. Returns the GPIB primary address of the instrument. Please refer to *To Change GPIB Address Settings* to learn how to set the GPIB address. Valid addresses are integers in the range of 1 to 30. (See page 1-2, *To Change GPIB Address Settings*.)

Group Bus

Syntax `GPIBUsb:ADDress?`

Returns `<NR1>` is the GPIB address of the power supply, as specified on the front-panel of the instrument. The range is from 1 to 30.

Examples `GPIBU:ADD?` might respond with 5, which is the GPIB address of the instrument.

GPIBUsb:ID? (Query Only)

For use in systems that include a TEK-USB-488 bus adapter module. Returns the identification string of a TEK-USB-488 bus adapter module.

Group Bus

Syntax `GPIBUsb:ID?`

Returns <manufacturer>, <model>, <serial number>, <firmware_version>

Examples `GPIBU:ID?`

*IDN? (Query Only)

Returns the power supply identification code in IEEE 488.2 notation.

Group Status

Syntax `*IDN?`

Returns A string that includes <manufacturer>, <model>, <serial number>, and <firmware_version> as defined in the following table.

<manufacturer>	<model>	<serial number>	<firmware_version>
tektronix	PWS4XXX	XXXXXX	X. XX-X. XX

Examples

```
*IDN?
```

might return the following response for a PWS4323:

```
TEKTRONIX , PWS4323 , 000004 , 1.01-1.20
```

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

This command initiates and executes a new current measurement, and returns the measured output current of the power supply.

Group Measurement

Syntax

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

Related Commands FETCh:VOLTage[:DC]?

Returns <NR2> is the measured output current in amperes.

Examples `MEAS:CURR?` might return 0.09998, which would be the measured current on the output of the power supply in amperes.

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

This command initiates and executes a new voltage measurement, and returns the measured output voltage of the power supply.

Group Measurement

Syntax

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

Related Commands FETCh:VOLTage[:DC]?

Returns <NR2> is the measured output voltage in volts.

Examples MEAS:VOLT? might return 5.0011 which would be the voltage measured across the power supply output in volts.

*OPC

This command configures the instrument to generate an operation complete message by setting bit 0 of the Standard Event Status Register (SESR) when all pending commands that generate an OPC message are complete.

The query command places the ASCII character "1" into the output queue when all such OPC commands are complete.

Group Synchronization

Syntax
```
*OPC
*OPC?
```

Examples *OPC? might return 1 to indicate that all pending OPC operations are finished.

*PSC

Sets and queries the power-on status flag that controls the automatic power-on execution of SRER and ESER. When *PSC is true, the SRER and ESER are set to 0 at power-on. When *PSC is false, the current values in the SRER and ESER are preserved in nonvolatile memory when power is shut off and are restored at power-on.

Group Source

Syntax
```
*PSC <NR1>
*PSC?
```

Related Commands *RST, *OPC

Arguments <NR1> = 0 sets the power-on status clear flag to false, disables the power-on clear, and allows the power supply to possibly assert SRQ after power on.

<NR1> ≠ 0 sets the power-on status clear flag to true. Sending *PSC 1 therefore enables the power-on status clear and prevents any SRQ assertion after power-on.

Returns 0|1

Examples *PSC 0
sets the power-on status clear flag to false.

*PSC?
might return 1, indicating that the power-on status clear flag is set to true.

*RCL (No Query Form)

Restores the state of the power supply from a copy of its settings stored in the setup memory. The settings are stored using the *SAV command. If the specified setup memory is deleted, this command causes an error.

Group Save and Recall

Syntax *RCL <NR1>

Related Commands *SAV

Arguments <NR1> is an integer value in the range from 0 to 40 and specifies the location of setup memory.

Examples *RCL 3
sets the power supply to settings stored in memory location 3.

*RST (No Query Form)

This command resets the power supply to default settings, but does not purge any stored settings.

Sending the *RST command does the following:

- Returns the power supply settings to the defaults. (See page C-1, *Default Setup*.)
- Clears the pending operation flag and associated operations

The *RST command does not change the following items:

- State of the USB or GPIB interface
- Calibration data that affects device specifications
- Current GPIB power supply address
- Stored settings
- Output queue
- Service Request Enable Register settings
- Standard Event Status Enable Register settings
- Power-On Status Clear flag setting
- front-panel LOCK state

Group Status

Syntax `*RST`

*SAV (No Query Form)

Saves the state of the power supply into a specified nonvolatile memory location. Any settings that had been stored previously at the location are overwritten. You can later use the *RCL command to restore the power supply to this saved state.

Group Status

Syntax `*SAV <NR1>`

Related Commands *RCL

Arguments `<NR1>` is an integer value in the range from 1 to 40.

Examples `*SAV 2`
saves the settings in memory location 2.

[SOURce:]CURRent[:LEVel]

This command sets the current value of the power supply in units of A or mA.

Group Source

Syntax
```
[SOURce:]CURRent[:LEVel] {<current>|MIN|MAX|DEF}
[SOURce:]CURRent[:LEVel]?
```

Arguments `<current>::=<NRf><units>`
where NRf is a flexible decimal between the minimum current and maximum nameplate current for the power supply. `<units>::={mA|A}`

`MIN` sets the current to the minimum level (0 A).

`MAX` sets the current to the maximum level.

`DEF` sets the current to the default level (0.1 A).

Returns `<NR2>` is the current setting in amperes.

Examples
```
CURR 3A
CURR 30mA
CURR MIN
```

`CURR?` might return 2.0000, which would be the current setting in amperes.

[SOURce:]DIGital:DATA

This command sets the output state of the rear-panel TTL control output and queries the state of the rear-panel TTL control input. When the port mode is DIGITAL, this command is enabled.

Group Source

Syntax
```
[SOURce:]DIGital:DATA <NR1>
[SOURce:]DIGital:DATA?
```

Arguments `<NR1>::={0|1}` sets the binary state of the rear-panel TTL control output to low or high state, respectively.

Returns `<NR1>::={0|1|2|3}` is a decimal representation of the state of the rear-panel TTL control output and input.

Value	Input Pin 9 & 10	Output Pin 11 & 12
0	Low	Low
1	Low	High
2	High	Low
3	High	High

Examples

```
DIGITAL:DATA ON
```

`DIGITAL:DATA?` might return 3, which would indicate that the TTL control input is high and the output is also high.

[SOURce:]DIGital:FUNCtion

This command sets or queries the function of the TTL control lines on the rear panel of the power supply.

*RST value is TRIGger.

Group Source

Syntax

```
[SOURce:]DIGital:FUNCtion {TRIGger|RIDFi|DIGital}
[SOURce:]DIGital:FUNCtion?
```

Related Commands TRIGger:SOURce, [SOURce:]OUTPut:DFI:SOURce, [SOURce:]DIGital:DATA

Arguments

Port mode	In	Out	Decription
`TRIGger`	Trigger In	N/A	Configures the TTL control input as an external trigger source.
`RIDFi`	RI	DFI	Configures the TTL control input as a remote inhibit (RI) input. The RI input can be used to turn the power supply output on or off from an external signal. Configures the TTL control output as a Discrete Fault Input (DFI). The DFI output provides a signal indicating a
`DIGital`	DI	DO	It can read and control the output port state by command. It configures the TTL input and output for direct control using the [SOURce:]DIGital:DATA command.

Returns `TRIG|RIDF|DIG`

Examples `DIGITAL:FUNCTION TRIGGER`

`DIG:FUNC?` might return RIDF, which would indicate that the TTL input and output are configure to function as remote inhibit and discrete fault input respectively.

[SOURce:]FUNCtion:MODE

This command configures the power supply to respond to discrete commands or to operate in list mode. Use this command with the FIXed parameter to exit list mode and prepare the instrument to respond to discrete settings changes.

NOTE. *When you modify any parameter in FIXed mode, the message "Setting conflict" will be returned. To avoid this, edit List files in FIXed mode and then switch to LIST mode before you run a List file.*

Group Source

Syntax
```
[SOURce:]FUNCtion:MODE {FIXed|LIST}
[SOURce:]FUNCtion:MODE?
```

Arguments `FIXed` configures the power supply to respond to discrete commands.

`LIST` configures the power supply to operate in list mode.

Returns `FIX|LIST`

Examples `[SOURCE:]FUNCTION:MODE LIST`

[SOURce:]LIST:COUNt

This command configures the number of times the active list will execute before stopping.

Group Source

Syntax

```
[SOURce:]LIST:COUNt {<NR1>|ONCE|REPeat}
[SOURce:]LIST:COUNt?
```

Related Commands [SOURce:]LIST:MODE, [SOURce:]LIST:STEP

Arguments

<NR1> is an integer between 2 and 65535. It determines the number of times the active list will execute.
ONCe: configures the lists to execute once and stop.
REPeat: configures the lists to loop continuously.

Returns <NR1> is the number of times the active list is set to execute.

Examples LIST:COUNT 6 would set the active list to execute 6 times before stopping.

[SOURce:]LIST:CURRent[:LEVel]

This command sets the current for a list step in units of A or mA.

Group Source

Syntax

```
[SOURce:]LIST:CURRent[:LEVel] <NR1>,<current>
[SOURce:]LIST:CURRent[:LEVel]?  <NR1>
```

Related Commands [SOURce:]LIST:VOLTage[:LEVel], [SOURce:]LIST:WIDth

Arguments

<NR1> is an integer in the range from 1 to 80, , which is a step number in the active list.

<current>::=<NRf>[<units]
where
<NRf> is a flexible decimal number used to specify a current setting in the range 0 amperes to the nameplate current rating of the power supply.
<units>::={A|mA}

Returns <NR2> is a decimal representing the current setting in amperes for the step specified in the query.

Examples

```
LIST:CURR 1, 3A
```

`LIST:CURR? 1` might respond with 3.0000, which indicates the current level setting for step number 1.

[SOURce:]LIST:MODE

This command determines the response of the power supply to a trigger in list mode.

Group Source

Syntax

```
[SOURce:]LIST:MODE {CONTinuous|STEP}
[SOURce:]LIST:MODE?
```

Related Commands [SOURce:]LIST:COUNt, [SOURce:]LIST:STEP

Arguments

`CONTinuous`: Sets the power supply to execute the entire list in response to a trigger.

`STEP`: Sets the power supply to execute one step per trigger.

Returns `CONT|STEP`

Examples

```
LIST:MODE CONT
```

`LIST:MODE?` might respond with STEP to indicate that the active list is configured to wait for one trigger for each step.

[SOURce:]LIST:RCL (No Query Form)

This command recalls a previously saved list from the specified storage location and makes it the active list for editing or execution.

Group Source: List

Syntax

```
[SOURce:]LIST:RCL <NR1>
```

Related Commands [SOURce:]LIST:SAVe

Arguments <NR1> is an integer in the range from 1 to 8, representing a list storage location.

Examples LIST:RCL 5 would make the list stored in location 5 the active list for editing or execution.

[SOURce:]LIST:SAVe (No Query Form)

This command saves the active list file to a storage location in non-volatile memory.

Group Source

Syntax
```
[SOURce:]LIST:SAVe <NR1>
```

Related Commands [SOURce:]LIST:RCL

Arguments <NR1> is an integer in the range from 1 to 8, representing a list storage location.

Examples LIST:SAV 4 saves the active list in list storage location 4.

[SOURce:]LIST:STEP

This command configures the number of steps in the active list. The number of steps must be configured before loading the voltage levels, current levels, and/or durations of the steps.

Group Source: List

Syntax
```
[SOURce:]LIST:STEP {<NR1>|MIN|MAX}
[SOURce:]LIST:STEP?
```

Related Commands [SOURce:]LIST:COUNt, [SOURce:]LIST:MODE

Arguments <NR1> is an integer in the range from 2 to 80, representing the number of steps to be configured in the list.

Returns <NR1> is the number of steps configured in the active list.

Examples Examples are provided at the end of this manual. (See page B-1, *Programming Examples*.)

[SOURce:]LIST:VOLTage[:LEVel]

This command sets the voltage level of a specified step in a list in units of V or mV.

Group Source: List

Syntax

```
[SOURce:]LIST:VOLTage[:LEVel] <NR1>,<voltage>
```

Arguments <NR1> is an integer in the range from 1 to 80, which is a step number in the active list.

<voltage>::=<NRf><units>
where
<NRf> is a flexible decimal that sets the voltage level fro the step.
<units>::={V|mV}

Returns <NR2>

Examples

```
LIST:VOLT 1, 3V
```

LIST:VOLT? 1 might return 3.0000, which would be the voltage level for step number 1.

[SOURce:]LIST:WIDth

This command sets the duration of a specified step in a list.

Group Source

Syntax

```
[SOURce:]LIST:WIDth <NR1>,{<duration>|MIN|MAX}
[SOURce:]LIST:WIDth?  <NR1>
```

Related Commands [SOURce:]LIST:VOLTage[:LEVel], [SOURce:]LIST:CURRent[:LEVel]

Arguments <NR1> is an integer in the range from 1 to 80, which is a step number in the active list. <duration>::=<NRf><units>
where

<NRf> is the time duration of the step
<units>::={s|ms}

Returns <NR2>

Examples LIST:WIDTH 12, 100ms

LIST:WIDTH? 12 might return 0.1, which would be the duration of step 12.

[SOURce:]OUTPut:DFI:SOURce

This command associates the DFI TTL output on the rear panel with a specified bit in the status byte register (SBR). Once the bit is associated with the DFI signal, the DFI signal will reflect the state of the specified bit. The port needs to be in the DFI or RI mode before using this command. Use the [SOURce:]DIGital:FUNCtion command to set the port mode.

*RST value is OFF.

Group Source: Digital I/O

Syntax
```
[SOURce:]OUTPut:DFI:SOURce {OFF|QUES|OPER|ESB|RQS}
[SOURce:]OUTPut:DFI:SOURce?
```

Related Commands [SOURce:]DIGital:FUNCtion

Arguments OFF: the output level of the DFI output pin remains high.

QUES: the output level of the DFI output pin reflects the complement of the state of the QUES bit. For example, when the QUES bit is 1, the DFI output pin is low.

OPER: the output level of the DFI output pin reflects the state of the OPER bit.

ESB: the output level of the DFI output pin reflects the state of the ESB bit.

RQS: the output level of the DFI output pin reflects the state of the RQS bit.

Returns OFF|QUES|OPER|ESB|RQS

Examples OUTput:DFI:SOURCE QUES sets the DFI signal to respond to the state of the QUES bit in the status byte register (SBR).

[SOURce:]OUTPut:PON[:STATe]

This command configures the power supply to power up with its output turned off, or to return the output to the state it was in when it powered down.

Group Source

Syntax

```
[SOURce:]OUTPut:PON[:STATe] {RST|RCL0}
[SOURce:]OUTPut:PON[:STATe]?
```

Arguments `RST` sets the power supply to power-up with output off.

`RCL0` sets the power supply to power-up with the output in the last state before power was removed.

Returns `RST|RCL0`

Examples

```
OUTPUT:PON RST
```

`OUTPUT:PON?` might return `RCL0`, which would indicate that the instrument will return to the present output state if the power is cycled.

[SOURce:]OUTPut:PROTection:CLEar (No Query Form)

This command clears a trip condition caused by over voltage (OV), over temperature (OT), or remote inhibit (RI).

Group Source

Syntax

```
[SOURce:]OUTPut:PROTection:CLEar
```

Related Commands [SOURce:]VOLTage:PROTection[:LEVel], [SOURce:]VOLTage:PROTection:STATe

Examples

```
OUTP:PROT:CLE
```

[SOURce:]OUTPut:RI:MODE

This command sets the input mode of the RI (remote inhibit) input pin. In order for this command to be effective, the rear panel TTL input and output must be configured in RI/DFI mode using the [SOURce:]DIGital:FUNCtion command.

Group Source

Syntax

```
[SOURce:]OUTPut:RI:MODE {OFF|LATChing|LIVE}
[SOURce:]OUTPut:RI:MODE?
```

Related Commands [SOURce:]DIGital:FUNCtion

Arguments `OFF`: The level of the RI input pin does not affect the output state of the power supply.

`LATChing`: When the level of the RI input pin changes from high to low, the output of power supply turns off.

`LIVE`: The output state of power supply changes according to the level of the RI input pin. While the level is TTL high, the output will be on. While the level is TTL low, the output of the power supply will be off.

Returns `OFF|LATC|LIVE`

Examples

```
OUTP:RI:MODE LATC
```

`OUTP:RI:MODE?` might return `OFF`, which would indicate that remote inhibit is not controlling the output of the instrument.

[SOURce:]OUTPut[:STATe]

This command turns the power supply output channel on or off.

Group Source

Syntax

```
[SOURce:]OUTPut[:STATe] {0|1|ON|OFF}
[SOURce:]OUTPut[:STATe]?
```

Related Commands [SOURce:]OUTPut:TIMer[:STATe]

Arguments

0 or OFF turns the power supply output off.

1 or ON turns the power supply output on.

Returns

1|0

Examples

```
OUTPUT ON
```

OUTPUT? might return 0, which would indicate that the output is off.

[SOURce:]OUTPut:TIMer:DELay

This command sets the time duration of the output timer. When the timer is activated and a duration is set, the output of the power supply will turn off automatically if left on longer than the specified duration. In order to ensure proper operation of the output timer, the timer must be activated using the [SOURce:]OUTPut:TIMer[:STATe] command before turning the output on.

Group

Source

Syntax

```
[SOURce:]OUTPut:TIMer:DELay {<duration>|MIN|MAX|DEF}
[SOURce:]OUTPut:TIMer:DELay?
```

Related Commands

[SOURce:]OUTPut:TIMer[:STATe], [SOURce:]OUTPut[:STATe]

Arguments

<duration> ::= <NRf><units>
where
<NRf> is a flexible decimal specifying time in the range 0.01s (or 10ms) to 60000s.
<units>::={S|ms}

MIN: The minimum time of the output timer (0.01 s).

MAX: The maximum time of the output timer (60,000 s).

DEF: The default time of the output timer (60 s).

Returns

<NR2> is the timer duration in seconds.

Examples

```
OUTP:TIM:DEL MIN
```

`OUTP:TIM:DEL?` might return `60.2`, which would represent the maximum time, in seconds, that the output of the instrument could be turned on if the timer is active.

[SOURce:]OUTPut:TIMer[:STATe]

This command turns the output timer function on and off. When the timer is activated and a duration is set, the output of the power supply will turn off automatically if left on longer than the specified duration. In order to ensure proper operation of the output timer, the timer must be activated using the [SOURce:]OUTPut:TIMer[:STATe] command before turning the output on.

Group Source

Syntax

```
[SOURce:]OUTPut:TIMer[:STATe] {0|1|ON|OFF}
[SOURce:]OUTPut:TIMer[:STATe]?
```

Related Commands [SOURce:]OUTPut:PROTection:CLEar, [SOURce:]OUTPut:TIMer:DELay, [SOURce:]OUTPut[:STATe]

Arguments `0` or `OFF` turns the output timer off.

`1` or `ON` turns the output timer on.

Returns `0|1`

Examples To start timer, first send `OUTPUT:TIMER:STATE ON`, then send `OUTPUT:STATE ON`.

To end timer (turn timer off), send `OUTPUT:TIMER:STATE OFF`.

[SOURce:]VOLTage[:LEVel]

This command sets the voltage value of the power supply.

Group Source

Syntax

```
[SOURce:]VOLTage[:LEVel] {<NRf>|MIN|MAX|DEF}
[SOURce:]VOLTage[:LEVel]?
```

Related Commands [SOURce:]CURRent[:LEVel]

Arguments <voltage>
where
<voltage>::=<NRf><units>
<NRf> is a flexible decimal specifying the voltage setting, ranging from 0 to the maximum nameplate voltage of the power supply.
<units>::={V|mV|kV}

MIN sets the voltage to the minimum level (0 V).

MAX sets the voltage to the maximum level (note that the maximum level may be somewhat higher than the nameplate).

DEF is the default level (1 V).

Returns NR2 is the voltage setting in volts.

Examples VOLTAGE:MIN

VOLTAGE? might return 1.05, which would be the voltage setting in volts.

[SOURce:]VOLTage:PROTection[:LEVel]

This command sets the over voltage protection (OVP) threshold level.

Group Source

Syntax
```
[SOURce:]VOLTage:PROTection[:LEVel] {<voltage>|MIN|MAX|DEF}
[SOURce:]VOLTage:PROTection[:LEVel]?
```

Related Commands [SOURce:]VOLTage:PROTection:STATe, [SOURce:]VOLTage:RANGe

Arguments <voltage>::=<NRf><units>
where
<NRf> is a flexible decimal that specifies the OVP threshold, ranging from 0 to the maximum OVP level
<units> ::={V|mV}

MIN: sets the minimum OVP level (1 V).

MAX: sets the maximum OVP level, which is approximately 10% higher than the maximum output voltage rating of the power supply.

DEF: sets the OVP level to the default level, which is equal to MAX.

Returns

<NR2> is the OVP threshold in volts.

MIN: set to minimum OVP level.

MAX: set to maximum OVP level.

Examples

```
VOLT:PROT 30V
```

VOLT:PROT? might return 30, which would be the OVP threshold in volts.

[SOURce:]VOLTage:PROTection:STATe

This command activates, deactivates, or checks the status of overvoltage protection (OVP).

Group

Source

Syntax

```
[SOURce:]VOLTage:PROTection:STATe {0|1|OFF|ON}
[SOURce:]VOLTage:PROTection:STATe?
```

Related Commands

[SOURce:]VOLTage:PROTection[:LEVel], [SOURce:]VOLTage:RANGe

Arguments

0 or OFF sets the over voltage protection to off.

1 or ON sets the over voltage protection to on.

Returns

0|1 means the over voltage protection is off.

Examples

```
VOLT:PROT:STAT 1
```

VOLT:PROT:STAT? might return 1, which would indicate that overvoltage protection is active.

[SOURce:]VOLTage:RANGe

This command limits the maximum voltage that can be programmed on the power supply. This command corresponds to the front-panel Max Voltage setting that can be found under the Protection submenu. This function is different from OVP, since it cannot turn the output off.

Group Source

Syntax
```
[SOURce:]VOLTage:RANGe {<voltage>|MIN|MAX|DEF}
[SOURce:]VOLTage:RANGe?
```

Related Commands [SOURce:]VOLTage:PROTection[:LEVel], [SOURce:]VOLTage:PROTection:STATe

Arguments `<voltage>::=<NRf><units>`
where
`<NRf>` is a flexible decimal that sets the maximum output voltage, ranging from 0 to the maximum nameplate voltage
`<units>::={V|mV|kV}`

`MIN`: sets the voltage limit to the minimum value (0 V).

`MAX`: sets the voltage limit to the maximum value which is slightly higher than the maximum nameplate voltage.

`DEF`: sets the voltage limit to the default value, which equals `MAX`.

Returns `<NR2>` is the voltage limit in volts.

Examples
```
VOLT:RANG 3.2V
```

`VOLT:RANG?` might return `3.2`, which would be the maximum programmable voltage in volts.

*SRE

(Service Request Enable) sets and queries the bits in the Service Request Enable Register (SRER). Refer to the *Status and Events* chapter for more information.

Group Status

Syntax
```
*SRE <NR1>
*SRE?
```

Related Commands *CLS, *ESR?, *PSC

Arguments `<NR1>` is an integer value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution

error. The power-on default for SRER is 0 if *PSC is 1. If *PSC is 0, the SRER maintains its value through a power cycle.

Examples

`*SRE 48` sets the bits in the SRER to 00110000 binary.

`*SRE?` might return a value of `32`, showing that the bits in the SRER have the binary value 00100000.

STATus:OPERation:CONDition? (Query Only)

This command returns the contents of the operation condition register (OCR). Details on the OCR are available in this manual. (See page 3-1, *Status and Events*.)

Group

Status

Syntax

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

Related Commands

STATus:OPERation:ENABle, STATus:OPERation[:EVENt]?

Returns

`<NR1>` is a decimal integer representation of the contents of the Operation Condition Register (OCR), ranging from 0 to 255.

Examples

`STATUS:OPERATION:CONDITION?` might return `4`, which would indicate that the power supply is in constant voltage mode.

STATus:OPERation:ENABle

This command sets and queries the contents of the operation enable register (OENR). The OENR is an eight-bit mask register that determines which bits in the Operation Event Register (OEVR) will affect the state of the OPER bit in the Status Byte Register (SBR). Details about the status registers are available in this manual. (See page 3-1, *Status and Events*.)

Group

Status

Syntax

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

Related Commands

STATus:OPERation:CONDition?, STATus:OPERation[:EVENt]?

Arguments

<mask>::=<NR1>
where
<NR1> is a decimal integer ranging from 0 through 255. The binary bits of the OENR are set according to this value.

Returns

<mask>

Examples

STATUS:OPERATION:ENABLE 8

STATUS:OPERATION:ENABLE? might return 8, which would indicate that only the Constant Current bit of the Operation Event Register would affect the OPER bit of the Status Byte Register.

STATus:OPERation[:EVENt]? (Query Only)

This command returns the contents of the operation event register (OEVR). After executing this command the operation event register is reset. Details about status registers are available in this manual. (See page 3-1, *Status and Events*.)

Group

Status

Syntax

STATus:OPERation[:EVENt]?

Related Commands

STATus:OPERation:CONDition?, STATus:OPERation:ENABle

Returns

<NR1> is a decimal integer representation of the contents of the Operation Event Register (OEVR), ranging from 0 to 255.

Examples

STATUS:OPERATION:EVENT? might return 10, which indicates that the power supply is waiting for trigger and is in a constant current mode.

STATus:QUEStionable:CONDition? (Query Only)

This command returns the contents of the questionable condition register (QCR). Details about the QCR are available in this manual. (See page 3-1, *Status and Events*.)

Group

Status

Syntax

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

Related Commands STATus:QUEStionable:ENABle, STATus:QUEStionable[:EVENt]?

Returns <NR1> is a decimal integer representation of the contents of the Questionable Condition Register (OCR), ranging from 0 to 255.

Examples STATUS:QUESTIONABLE:CONDITION? might return 1, which would indicate an over voltage condition.

STATus:QUEStionable:ENABle

This command sets and queries the contents of the questionable enable register (QENR). The QENR is an eight-bit mask register that determines which bits in the Questionable Event Register (QEVR) will affect the state of the QUES bit in the Status Byte Register (SBR).

Group Status

Syntax

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

Related Commands STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:CONDition?, STATus:QUEStionable:PTRansition, STATus:QUEStionable:NTRansition, *PSC

Arguments

<mask>::=<NR1>
where
<NR1> is a decimal integer ranging from 0 through 255. The binary bits of the QENR are set according to this value.

Returns <mask>

Examples STATUS:QUESTIONABLE:ENABLE 8

STATUS:QUESTIONABLE:ENABLE? might return 8, which would indicate that only a transition of the Remote Inhibit bit of the QCR would affect the QUES bit of the Status Byte Register.

STATus:QUEStionable[:EVENt]? (Query Only)

This command returns the contents of the questionable event register (QEVR). After executing this command, the quest event register is reset. Details about the QEVR are available in this manual. (See page 3-1, *Status and Events*.)

Group Status

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

Related Commands STATus:QUEStionable:ENABle, STATus:QUEStionable:CONDition?, STATus:QUEStionable:PTRansition, STATus:QUEStionable:NTRansition

Returns <NR1> is a decimal integer representation of the contents of the Questionable Event Register (QEVR), ranging from 0 to 255.

Examples `STATUS:QUESTIONABLE:EVENT?` might return `1`, which would indicate an overvoltage condition.

STATus:QUEStionable:NTRansition

This command sets the negative transition filter of the questionable event register. The filter contents cause the corresponding bit in the questionable event register to become 1 when the bit value of the questionable condition register transitions from 1 to 0.

Group Status

Syntax
```
STATus:QUEStionable:NTRansition <mask>
STATus:QUEStionable:NTRansition?
```

Related Commands STATus:QUEStionable:ENABle, STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:CONDition?, STATus:QUEStionable:PTRansition

Arguments
```
<mask>::=<NR1>
```
where
<NR1> is a number ranging from 0 through 255. The binary bits of the Questionable NTR register are set according to this value.

Returns <mask>

Examples STATUS:QUESTIONABLE:NTRANSITION 8

STATUS:QUESTIONABLE:NTRANSITION? might return 8, which would indicate that a negative transition on the Remote Inhibit bit of the QCR would be registered as an event.

STATus:QUEStionable:PTRansition

This command sets the positive transition filter of the questionable event register. The filter contents cause the corresponding bit in the questionable event register to become 1 when the bit value of the questionable condition register transitions from 0 to 1.

Group Status

Syntax
```
STATus:QUEStionable:PTRansition <mask>
STATus:QUEStionable:PTRansition?
```

Related Commands STATus:QUEStionable:ENABle, STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:CONDition?, STATus:QUEStionable:NTRansition

Arguments
<mask>::=<NR1>
where
<NR1> is a number ranging from 0 through 255. The binary bits of the Questionable PTR register are set according to this value.

Returns <mask>

Examples STATUS:QUESTIONABLE:PTRANSITION 8

STATUS:QUESTIONABLE:PTRANSITION? might return 8 which would indicate that a positive transition on the Remote Inhibit bit of the QCR would be registered as an event.

*STB? (Query Only)

The byte query returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. Refer to the *Status and Events* chapter for more information. (See page 3-3.)

Group Status

Syntax `*STB?`

Related Commands *ESE, *CLS, *ESR?

Returns `<NR1>`

Examples `*STB?` 96

shows that the SBR contains the binary value 01100000.

SYSTem:ERRor? (Query Only)

This command queries the error code and error information of the power supply and returns both values. (See Table 3-6 on page 3-9.)

Group System

Syntax `SYSTem:ERRor?`

Returns
```
<NR1>,<error_text>
<error_text> ::= <string>
```
where `<string>` is a description of the error.

Examples `SYSTEM:ERROR?` might return `110`, which means No Input Command to parse.

SYSTem:KEY

This command can produce the same effect as pressing one of the front-panel buttons. The instrument must be in local mode in order for this command to simulate a front-panel button press.

Group System

Syntax
```
SYSTem:KEY <NR1>
SYSTem:KEY?
```

Arguments <NR1> is an integer key code (see the following table).

Returns <NR1>

Front-panel button	<NR1> key code
KEY_VSET	1
KEY_ISET	2
KEY_SAVE	3
KEY_RECALL	4
KEY_LEFT	5
KEY_RIGHT	6
KEY_UP	7
KEY_DOWN	8
KEY_0	9
KEY_1	10
KEY_2	11
KEY_3	12
KEY_4	13
KEY_5	14
KEY_6	15
KEY_7	16
KEY_8	17
KEY_9	18
KEY_DECIMAL	19
KEY_ESC	20
KEY_ENTER	21
KEY_ON	22
KEY_SHIFT	64

Examples SYSTEM:KEY 64 would simulate a press of the Shift key.

SYSTem:LOCal (No Query Form)

This command sets the power supply for control from the front-panel.

Group System

Syntax SYSTem:LOCal

Related Commands SYSTem:REMote, SYSTem:RWLock

Examples

```
SYS:LOC
```

SYSTem:POSetup

This command determines how the power supply initializes when its power switch is turned on. This command configures the instrument to power up with default settings, or power up with the settings that were in effect when the instrument was turned off.

Group System

Syntax

```
SYSTem:POSetup {RST|RCL0}
SYSTem:POSetup?
```

Arguments

`RST`: initializes the power supply to default settings after a power cycle.

`RCL0`: saves the most recent settings and restores these after a power cycle.

Returns

`RST`: default settings are applied after a power cycle.

`RCL0`: most recent settings are saved and restored after a power cycle.

Examples

```
SYST:POS RST
```

`SYSTEM:POSETUP?` might respond with `RST`, which would indicate that the power supply is configured to restore the power supply to default settings when it powers up.

SYSTem:REMote (No Query Form)

This command sets the power supply to remote control mode.

Group System

Syntax

```
SYSTem:REMote
```

Related Commands SYSTem:LOCal, SYSTem:RWLock

Arguments None.

Examples

```
SYSTEM:REMOTE
```

SYSTem:RWLock (No Query Form)

If the power supply is in remote mode, this command locks out the front panel LOCAL button. This command has no effect if the instrument is in local mode.

Group System

Syntax

```
SYSTem:RWLock
```

Related Commands SYSTem:REMote, SYSTem:LOCal

Arguments None.

Examples

```
SYSTEM:RWLOCK
```

SYSTem:VERSion? (Query Only)

This command returns SCPI version of the instrument.

Group System

Syntax

```
SYSTem:VERSion?
```

Returns <NR2> is the software version of the power supply.

Examples `SYSTEM:VERSION?` might return `1991.0`, which is the SCPI version number.

*TRG (No Query Form)

This command generates a trigger event.

Group Trigger

Syntax `*TRG`

Related Commands TRIGger[:IMMediate]

Examples `*TRG`

TRIGger[:IMMediate] (No Query Form)

This command forces an immediate trigger event.

Group Trigger

Syntax `TRIGger[:IMMediate]`

Related Commands *TRG

Arguments None.

Examples `TRIGGER`

TRIGger:SOURce

This command sets the source of trigger events.

Group Trigger

Syntax
```
TRIGger:SOURce {MANual|IMMediate|EXTernal|BUS}
TRIGger:SOURce?
```

Related Commands [SOURce:]DIGital:FUNCtion, *TRG, TRIGger[:IMMediate]

Arguments `MANual`: When this value is sent, the power supply will wait for a trigger from the front-panel. Press Shift + 3 (Trigger) to trigger the power supply.

IMMediate: When this value is sent, the power supply will wait for a TRIgger:IMMediate bus command.

EXTernal: When this value is sent, the power supply can be triggered with a TTL pulse applied to pin 1 of the terminal connector in the rear. The pulse width should be at least 5 ms.

BUS: When this value is sent, the power supply can be triggered by sending a *TRG or TRIgger:IMMediate command to the power supply.

Examples

```
TRIGGER:SOURCE BUS
```

*TST? (Query Only)

Initiates a self-test and reports any errors.

Group Diagnostic

Syntax

```
*TST?
```

Returns

<NR1>
where
<NR1>= 0 indicates that the self-test completed with no errors.
<NR1> not equal to 0 indicates that the self test detected an error.

Self test code descriptions are available. (See Table 3-10 on page 3-10.)

*WAI (No Query Form)

This command prevents the instrument from executing further commands or queries until all pending commands are complete.

Group Synchronization

Syntax

```
*WAI
```

Examples

```
*WAI
```

Status and Events

Status and Events

This section provides details about the status information and events the power supply reports.

Status Reporting Structure

A diagram is provided showing an outline of the power supply error and event reporting function. (See Figure 3-1.)

The error and event reporting system consists of the following three blocks:

- Standard/Event Status
- Operation Status
- Questionable Status

The operations processed in these blocks are summarized in status bytes, which provide the error and event data.

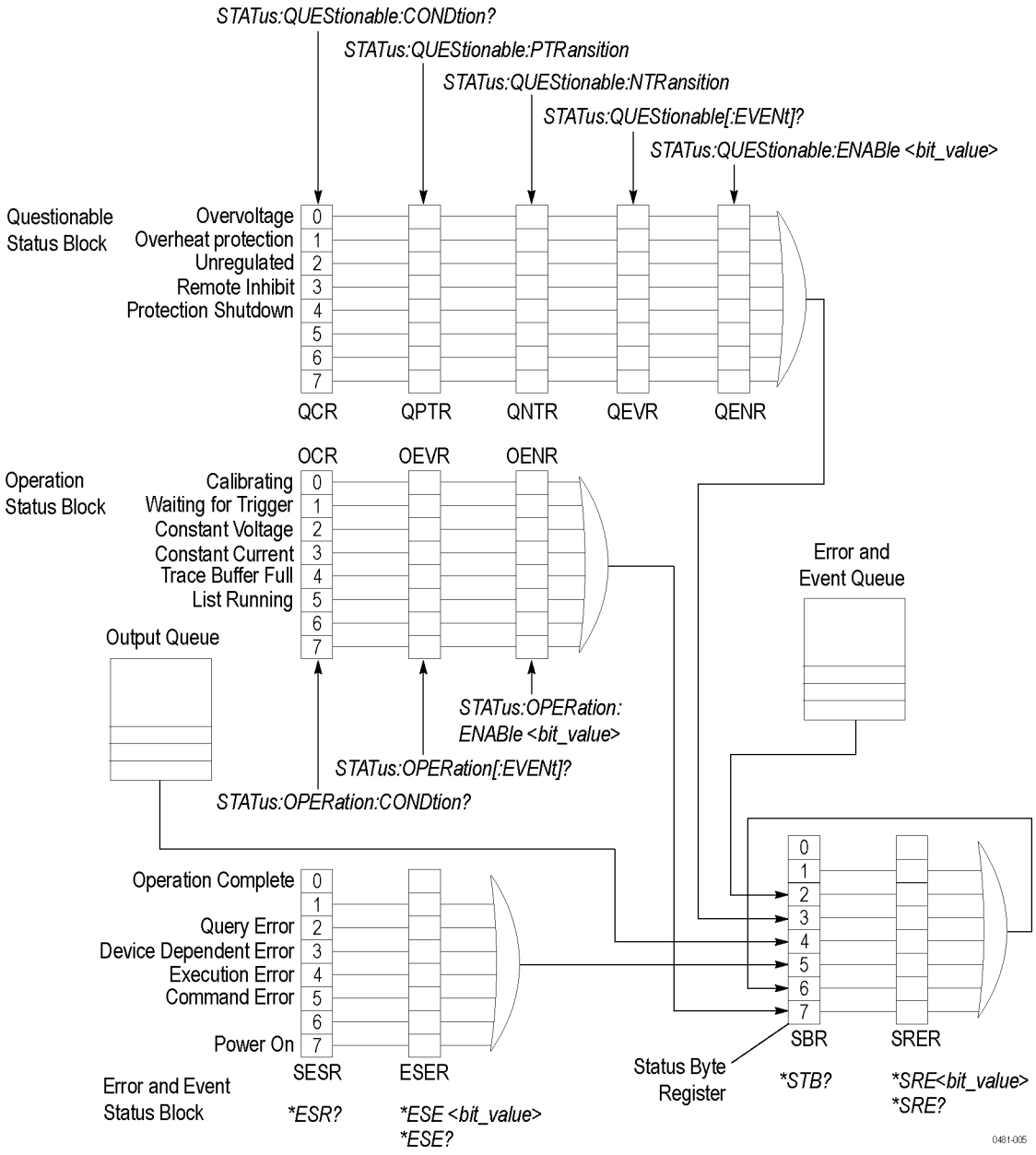


Figure 3-1: Error and event handling process

Registers

The registers in the event reporting system fall into two functional groups:

- Status Registers contain information about the status of the power supply. They include the Standard Event Status Register (SESR).
- Enable Registers determine whether selected types of events are reported to the Status Registers and the Event Queue. They include the Event Status Enable Register (ESER), the Service Request Enable Register (SRER), the Operation Enable Register (OENR), and the Questionable Enable Register (QENR).

Status Registers

There are six types of status registers:

- Status Byte Register (SBR). (See page 3-3.)
- Standard Event Status Register (SESR). (See page 3-4.)
- Operation Condition Register (OCR). (See page 3-5.)
- Operation Event Register (OEVR). (See page 3-5.)
- Questionable Condition Register (QCR). (See page 3-5.)
- Questionable Event Register (QEVR). (See page 3-6.)

NOTE. *The Questionable Event Register may be filtered by a positive transition filter (QPTR) and a negative transition filter (QNTR). (See page 3-6.)*

The Status Byte Register (SBR). The SBR is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2-1992. These bits are used to monitor the output queue, SESR, and service requests, respectively. (See Figure 3-2.)

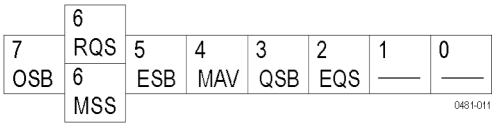


Figure 3-2: The Status Byte Register (SBR)

Table 3-1: SBR bit functions

Bit	Function	
7 (MSB)	OSB	`Operation Status Bit.` Indicates that an operation event has occurred.
6	RQS	`Request Service.` Obtained from a serial poll. Shows that the power supply requests service from the GPIB controller.

Table 3-1: SBR bit functions (cont.)

Bit	Function	
6	MSS	`Master Status Summary.` Obtained from *STB? query. Summarizes the OSB, ESB, MAV, QSB, and EQS bits in the SBR.
5	ESB	`Event Status Bit.` Shows that status is enabled and present in the SESR.
4	MAV	`Message Available.` Shows that output is available in the Output Queue.
3	QSB	`Questionable Status Bit.` Indicates that a questionable event has occurred.
2	EQS	Shows that information is available in the Error/Event Queue.
1	———	Not used.
0	———	Not used.

The Standard Event Status Register (SESR). The SESR records six types of events that can occur within the power supply as shown in the following figure. (See Figure 3-3.)

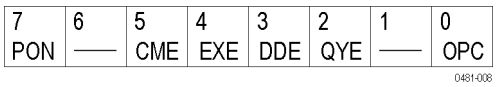


Figure 3-3: The Standard Event Status Register (SESR)

Table 3-2: SESR bit functions

Bit	Function	
7 (MSB)	PON	`Power On.` Shows that the power supply was powered on.
6	———	`User Request.` This bit is not used.
5	CME	`Command Error.` Shows that an error occurred while the power supply was parsing a command or query.
4	EXE	`Execution Error.` Shows that an error occurred while the power supply was executing a command or query.
3	DDE	`Device Error.` Shows that a device dependent error occurred.
2	QYE	`Query Error.` Either an attempt was made to read the Output Queue when no data was present or pending, or that data in the Output Queue was lost.
1	———	`Request Control.` This bit is not used.
0 (LSB)	OPC	`Operation Complete.` Shows that the operation is complete. This bit is set when all pending operations complete following an *OPC command.

The Operation Condition Register (OCR). The Operation Condition Register is made up of eight bits, which note the occurrence of events as shown here.

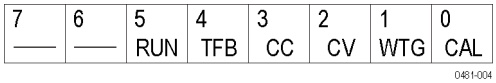


Figure 3-4: The Operation Condition Register (OCR)

Table 3-3: OCR bit functions

Bit	Function	
7 (MSB)	———	This bit is not used.
6	———	This bit is not used.
5	RUN	`Running List.` Indicates that the power supply is executing a list.
4	TBF	`Trace Buffer Full.` This bit is used internally, by the instrument.
3	CC	`Constant Current.` Indicates that the power supply is in constant current mode and is regulating its output current.
2	CV	`Constant Voltage.` Indicates that the power supply is in constant voltage mode and is regulating its output voltage.
1	WTG	`Waiting for Trigger.` Indicates that the power supply is waiting for a trigger.
0 (LSB)	CAL	`Calibrating.` Indicates that the power supply is calculating a new calibration parameter.

The Operation Event Register (OEVR). The Operation Event Register has the same content as the Operation Condition Register.

The Questionable Condition Register (QCR). The Questionable Condition Register is made up of eight bits which note the occurrence of five types of events as shown here.

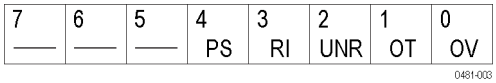


Figure 3-5: The Questionable Condition Register (QCR)

Table 3-4: QCR bit functions

Bit	Function	
7 (MSB)	———	This bit is not used.
6	———	This bit is not used.
5	———	This bit is not used.
4	PS	`Protection Shutdown.`
3	RI	`Remote Inhibit.` This bit indicates the state of the Remote Inhibit input.

Table 3-4: QCR bit functions (cont.)

Bit	Function	
2	UNR	`Unregulated.` Indicates that the power supply has gone out of regulation.
1	OT	`Over Temperature.` Indicates that the power supply has experienced an over temperature condition and the output has been shut down.
0 (LSB)	OV	`Over Voltage.` Indicates that the overvoltage protection threshold has been exceeded and the output has shut down.

The Questionable Positive Transition Register (QPTR) and Questionable Negative Transition Register (QNTR). The QEVR can be configured to respond to specific transitions in the QCR. The transitions and the polarity of these transitions are specified in the QPTR and QNTR. These registers act as filters between the QCR and the QEVR. If a bit is set to 1 in the QNTR, the corresponding bit in the QEVR will be set to 1 whenever the same bit in the QCR makes a negative transition, from 1 to 0. If a bit is set to 1 in the QPTR, the corresponding bit in the QEVR will be set to 1 when the same bit in the QCR makes a positive transition, from 0 to 1.

The Questionable Event Register (QEVR). The Questionable Event Register is made up of eight bits, which have the same significance as the QCR, but have been filtered by the QPTR and QNTR.

Enable Registers

There are four types of enable registers:

- Event Status Enable Register (ESER), (See page 3-7.)
- Service Request Enable Register (SRER), (See page 3-7.)
- Operation Enable Register (OENR), (See page 3-7.)
- Questionable Enable Register (QENR), (See page 3-7.)

ESER, SRER, OENR, and QENR allow you to select which events are reported to the Status Byte Register (SBR).

Each bit in an Enable Register corresponds to a bit in an Event Register. In order for an event to be reported to a bit in the Status Byte Register, the corresponding bit in the Enable Register must be set to one. If the bit in the Enable Register is set to zero, the event will not affect the status bit.

Various commands set the bits in the Enable Registers. The Enable Registers and the commands used to set them are described below.

The Event Status Enable Register (ESER). This register controls which types of events are summarized by the Event Status Bit (ESB) in the SBR. 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	——	CME	EXE	DDE	QYE	——	OPC

0481-010

Figure 3-6: The Event Status Enable Register (ESER)

The Service Request Enable Register (SRER). This register controls which bits in the SBR generate a Service Request and are summarized by the Master Status Summary (MSS) bit.

Use the *SRE command to set the SRER. Use the *SRE? query to read the register. The RQS bit remains set to one until either the Status Byte Register is read with a Serial Poll or the MSS bit changes back to a zero.

	6						
7	RQS	5	4	3	2	1	0
OSB	6	ESB	MAV	QSB	EAV	——	——
	MSS						

0481-009

Figure 3-7: The Service Request Enable Register (SRER)

The Operation Enable Register (OENR). The OENR consists of bits defined exactly the same as bits 0 through 7 in the OEVR. You can use this register to control whether or not the Operation Status Bit (OSB) in the SBR is set when an event occurs and the corresponding OEVR bit is set.

Use the STATus:OPERation:ENABle command to set the bits in the OENR.

Use the STATus:OPERation:ENABle? query to read the contents of the OENR.

7	6	5	4	3	2	1	0
——	——	RUN	TFB	CC	CV	WTG	CAL

0481-004

Figure 3-8: The Operation Enable Register (OENR)

The Questionable Enable Register (QENR). The QENR consists of bits defined exactly the same as bits 0 through 7 in the QEVR register. You can use this register to control whether the QSB in the SBR is set when an event occurs and the corresponding QEVR bit is set.

Use the STATus:QUEStionable:ENABle command to set the bits in the QENR

Use the STATus:QUEStionable:ENABle? query to read the contents of the QENR

7	6	5	4	3	2	1	0
——	——	——	PS	RI	UNR	OT	OV

0481-003

Figure 3-9: The Questionable Enable Register (QENR)

*PSC Command

The *PSC command controls the Enable Registers contents at power-on. Sending *PSC 1 sets the Enable Registers at power on as follows:

- ESER 0 (equivalent to an *ESE 0 command)
- SRER 0 (equivalent to an *SRE 0 command)

Sending *PSC 0 lets the Enable Registers maintain their values in nonvolatile memory through a power cycle.

Queues

Output Queue

The power supply stores query responses in the Output Queue and empties this queue each time it receives a new command or query message after an <EOM>. The controller must read a query response before it sends the next command (or query) or it will lose responses to earlier queries.

Error/Event Queue

The Event Queue stores detailed information on up to 32 events. When 32 events stack up in the Event Queue, the 32nd event is replaced by event code 350, "Queue Overflow."

Read the Event Queue with the EVENT? query (which returns only the event number), with the EVMSG? query (which returns the event number and a text description of the event), or with the ALLEV? query (which returns all the event numbers with a description of the event). Reading an event removes it from the queue.

Before reading an event from the Event Queue, you must use the *ESR? query to read the summary of the event from the SESR. This makes the events summarized by the *ESR? read available to the EVENT? and EVMSG? queries, and empties the SESR.

Reading the SESR erases any events that were summarized by previous *ESR? reads but not read from the Event Queue. Events that follow an *ESR? read are put in the Event Queue but are not available until *ESR? is used again.

Messages and Codes

Error and event codes with negative values are SCPI standard codes. Error and event codes with positive values are unique to the PWS4000 Series Linear DC Power Supplies.

Table 3-5: No event messages

Code	Message
0	No events to report; queue empty
1	No events to report; new events pending *ESR?

Command Errors

The following table shows the command error messages generated by improper syntax. Check that the command is properly formed and that it follows the rules in the section on command Syntax.

Table 3-6: Command error messages (CME bit 5)

Code	Message
101	Design error: Too many numeric suffices in Command Spec
110	No Input Command to parse
114	Numeric suffix is invalid value
116	Invalid value in numeric or channel list, e.g. out of range
117	Invalid number of dimensions in a channel list
120	Parameter of type Numeric Value overflowed its storage
130	Wrong units for parameter
140	Wrong type of parameter(s)
150	Wrong number of parameters
160	Unmatched quotation mark in parameters (single/double)
165	Unmatched bracket
170	Command keywords were not recognized
180	No entry in list to retrieve
190	Too many dimensions in entry to be returned in parameters
191	Too many char

Execution Errors

The following table lists the execution errors that are detected during execution of a command.

Table 3-7: Execution error messages (EXE bit 4)

Code	Message
–200	Execution error
–221	Settings conflict
–222	Data out of range
–223	Too much data
–224	Illegal parameter value
–225	Out of memory
–270	Macro error
–272	Macro execution error
–273	Illegal macro label
–276	Macro recursion error
–277	Macro redefinition not allowed

System Errors

The following table lists the system errors that can occur during power supply operation.

Table 3-8: System error messages (DDE bit 3)

Code	Message
–310	System error
–350	Too many errors

Query Errors

The following table lists the query errors that can occur during power supply operation. These errors may indicate that there was a problem during the query process and that your query will not be performed.

Table 3-9: Query error messages (Standard Event Status Register bit 2)

Code	Message
–400	Query error
–410	Query INTERRUPTED
–420	Query UNTERMINATED
–430	Query DEADLOCKED
–440	Query UNTERMINATED

Self Test Errors

The following table lists the self test errors that can occur during power supply operation.

Table 3-10: Self test error messages (Standard Event Status Register bit 3)

Code	Message
0	No error
1	Module Initialization Lost
2	Mainframe Initialization Lost
3	Module Calibration Lost
4	Non-volatile RAM STATE section checksum failed
5	Non-volatile RAM RST section checksum failed
10	RAM selftest
40	Flash write failed
41	Flash erase failed
80	Digital I/O selftest error

Device Dependent Errors

The following table lists the device errors that can occur during power supply operation. These errors may indicate that the power supply needs repair.

Table 3-11: Device dependent error messages (DDE bit 3)

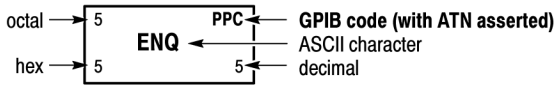
Code	Message
220	Front panel uart overrun
221	Front panel uart framing
222	Front panel uart parity
223	Front panel buffer overrun
224	Front panel timeout
225	Front Crc Check error
226	Front Cmd Error
401	CAL switch prevents calibration
402	CAL password is incorrect
403	CAL not enabled
404	Computed readback cal constants are incorrect
405	Computed programming cal constants are incorrect
406	Incorrect sequence of calibration commands
407	CV or CC status is incorrect for this command
603	FETCH of data that was not acquired
604	Measurement overrange

Appendices

Appendix A: ASCII Code Chart

B7 B6 B5 BITS B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 LA0 SP 20 32	60 LA16 0 30 48	100 TA0 @ 40 64	120 TA16 P 50 80	140 SA0 ` 60 96	160 SA16 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LL0 DC1 11 17	41 LA1 ! 21 33	61 LA17 1 31 49	101 TA1 A 41 65	121 TA17 Q 51 81	141 SA1 a 61 97	161 SA17 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 LA2 " 22 34	62 LA18 2 32 50	102 TA2 B 42 66	122 TA18 R 52 82	142 SA2 b 62 98	162 SA18 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 LA3 # 23 35	63 LA19 3 33 51	103 TA3 C 43 67	123 TA19 S 53 83	143 SA3 c 63 99	163 SA19 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 LA4 $ 24 36	64 LA20 4 34 52	104 TA4 D 44 68	124 TA20 T 54 84	144 SA4 d 64 100	164 SA20 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 LA5 % 25 37	65 LA21 5 35 53	105 TA5 E 45 69	125 TA21 U 55 85	145 SA5 e 65 101	165 SA21 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 LA6 & 26 38	66 LA22 6 36 54	106 TA6 F 46 70	126 TA22 V 56 86	146 SA6 f 66 102	166 SA22 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 LA7 ' 27 39	67 LA23 7 37 55	107 TA7 G 47 71	127 TA23 W 57 87	147 SA7 g 67 103	167 SA23 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 LA8 (28 40	70 LA24 8 38 56	110 TA8 H 48 72	130 TA24 X 58 88	150 SA8 h 68 104	170 SA24 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51 LA9) 29 41	71 LA25 9 39 57	111 TA9 I 49 73	131 TA25 Y 59 89	151 SA9 i 69 105	171 SA25 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 LA10 * 2A 42	72 LA26 : 3A 58	112 TA10 J 4A 74	132 TA26 Z 5A 90	152 SA10 j 6A 106	172 SA26 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 LA11 + 2B 43	73 LA27 ; 3B 59	113 TA11 K 4B 75	133 TA27 [5B 91	153 SA11 k 6B 107	173 SA27 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 LA12 , 2C 44	74 LA28 < 3C 60	114 TA12 L 4C 76	134 TA28 \ 5C 92	154 SA12 l 6C 108	174 SA28 ¦ 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 LA13 – 2D 45	75 LA29 = 3D 61	115 TA13 M 4D 77	135 TA29] 5D 93	155 SA13 m 6D 109	175 SA29 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 LA14 . 2E 46	76 LA30 > 3E 62	116 TA14 N 4E 78	136 TA30 ^ 5E 94	156 SA14 n 6E 110	176 SA30 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 LA15 / 2F 47	77 UNL ? 3F 63	117 TA15 O 4F 79	137 UNT – 5F 95	157 SA15 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY



Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Appendix B: Programming Examples

Example 1

This example is written in the C programming language; TekVISA or NIVISA can be used. It demonstrates basic communication with the power supply and error checking. The program establishes communication with the power supply, and puts it into remote mode. It then initializes the voltage and current and turns the output on. It sends new values for the voltage and current, and reads back the actual meter values before turning off the power supply output and closing communications.

```
// pws_visa.cpp
//

#include "stdafx.h"
#include <visa.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <conio.h>
#include <stdlib.h>

ViSession defaultRM; //Resource manager id
ViSession PWS4000; //Identifies the power supply
long ErrorStatus;
char commandString[256];
char ReadBuffer[256];

void OpenPort();
void SendSCPI(char* pString);
void CheckError(char* pMessage);
void delay(clock_t wait);
void ClosePort();

int main(int argc, _TCHAR* argv[])
{
  double voltage;
  char Buffer[256];
  double current;

  OpenPort();

  //Query the power supply id, read response and print it
  sprintf(Buffer, "*IDN?");
  SendSCPI(Buffer);
  printf("Instrument identification string:%s \n", Buffer);

  SendSCPI("*RST"); //reset the power supply
  SendSCPI("CURRENT 0.1A"); //set the current to 0.1A
  SendSCPI("VOLTAGE 3V"); //set the voltage to 3V
  SendSCPI("OUTPUT 1"); // turn output on
```

```
  voltage=5.0;
  current=0.2;
  printf("Setting voltage(V) & current(A): %f,%f \n",
voltage, current);

  ErrorStatus = viPrintf(PWS4000,"VOLT %f\n",voltage); //set
the output voltage
  CheckError("Unable to set voltage");
  ErrorStatus = viPrintf(PWS4000,"CURRENT %f\n",current);
//set the output current
  CheckError("Unable to set current");

  ErrorStatus = viPrintf(PWS4000,"MEASURE:VOLTAGE?\n");
//measure the output voltage
  CheckError("Unable to write the device");
  ErrorStatus = viScanf(PWS4000,"%f",&voltage); //retrieve
reading
  CheckError("Unable to read voltage");

  ErrorStatus = viPrintf(PWS4000,"MEASURE:CURRENT?\n");
//measure the output current
  CheckError("Unable to write the device");
  ErrorStatus = viScanf(PWS4000,"%f",&current); //retrieve
reading
  CheckError("Unable to read current");

  printf("Measured voltage(V) & current(A): %f,%f \n",
voltage, current);

SendSCPI("OUTPUT 0"); //turn output off
ClosePort();

  while(1);
  //return 0;
}

void OpenPort()
{

  //Open communication session with the power supply, and
put the power supply in remote
  ErrorStatus = viOpenDefaultRM(& defaultRM);
    ErrorStatus =viOpen(defaultRM,
"USB0::0X0699::0X0391::006002206573001001::INSTR",0,0,&PWS4000);
 CheckError("Unable to open the port");

  SendSCPI("SYSTEM:REMOTE");

}

void SendSCPI(char* pString)
{
```

```
 char* pdest;

 strcpy(commandString,pString);
 strcat(commandString, "\n");
 ErrorStatus = viPrintf(PWS4000, commandString);
 CheckError("Can't Write to Power Supply");

 pdest = strchr(commandString, '?'); //Search for query
command
 if (pdest != NULL)
 {

 ErrorStatus = viBufRead(PWS4000, (ViBuf)ReadBuffer,
sizeof(ReadBuffer), VI_NULL);
 CheckError("Can't read from driver");
 strcpy(pString, ReadBuffer);
 }
}

void ClosePort()
{

  viClose(PWS4000);

  viClose(defaultRM);
}

void CheckError(char* pMessage)

{

if(ErrorStatus != VI_SUCCESS)
  {

  printf("\n %s",pMessage);

  ClosePort();

  exit(0);

  }

}

void delay(clock_t wait)
 {

 clock_t goal;
 goal = wait + clock();
 while(goal > clock());

}
```

Example 2 This example shows a command sequence that configures a three-step list to execute once in continuous mode. It saves the list in location 1, and then places the instrument in list mode and runs the list.

```
SYSTEM:REMOTE
*RST
TRIGGER:SOURCE BUS
LIST:MODE CONT
LIST:COUNT ONCE
LIST:STEP 3
LIST:VOLT 1,2V
LIST:CURRENT 1,1.0A
LIST:WIDTH 1,5s
LIST:VOLT 2,4V
LIST:CURRENT 2,1.0A
LIST:WIDTH 2,10s
LIST:VOLT 3,0V
LIST:CURRENT 3,0.1A
LIST:WIDTH 3,2s
LIST:SAVE 1
FUNCTION:MODE LIST
OUTPUT ON
TRIGGER IMMEDIATE
```

Example 3 This example shows a command sequence that configures a two-step list to execute 20 times. It saves the list in location 1, and then places the instrument in list mode and runs the list.

```
SYSTEM:REMOTE
*RST
TRIGGER:SOURCE BUS
LIST:MODE CONT
LIST:COUNT 20
LIST:STEP 2
```

```
LIST:VOLTAGE 1,5V
LIST:CURRent 1,1.0A
LIST:WIDTH 1,10s
LIST:VOLTAGE 2,5.5V
LIST:CURRent 2,1.0A
LIST:WIDTH 2,5s
LIST:SAVE 1
FUNCTION:MODE LIST
OUTPUT ON
TRIGGER:IMMEDIATE
```

Example 4 This example is written in the C programming language; TekVISA or NIVISA can be used. The program demonstrates setting up and executing a list. It shows how to use the Operation Condition Register to determine when the list has finished executing.

```
// pws_visa.c
//

#undef MICROSOFT // Change this to #define if Microsoft
#ifdef MICROSOFT
 #include “stdafx.h”
 #include <conio.h>
#else
 typedef char_TCHAR;
#endif

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

ViSession defaultRM; //Resource manager id
ViSession PWS4000; //Identifies the power supply
long ErrorStatus;
char commandString[256];
char ReadBuffer[256];

//
// Some compilers do not require that void arguments be
declared, however, all should accept declaration of
// void argument lists.
//
```

```
void OpenPort(); // void added
void SendSCPI(char* pString);
void CheckError(char* pMessage);
void delay(clock_t wait);
void ClosePort(); // void added

int main(int argc, _TCHAR* argv[])
  {
  double voltage;
  char Buffer[256];
  double current;
  int OPERreg,i;
  float meas_voltage, meas_current;
  int List_count=10;
  int List_step =3;
  double List_Volt[3]= {3.0, 4.0, 6.0};
  double List_Curr[3] = {0.1, 0.1, 0.1};
  int List_Time[3]= {1,2,3};
  }

  OpenPort();

  //Query the power supply id, read response and print it
  sprintf(Buffer, "*IDN?");
  SendSCPI(Buffer);
  printf("Instrument identification string:%s \n", Buffer);

  SendSCPI("*RST"); //reset the power supply
  delay(100)

  SendSCPI("CURRENT 0.1A"); //set the current to 0.1A
  SendSCPI("VOLTAGE 0V"); //set the voltage to 0V
  SendSCPI("OUTPUT 1"); // turn output on

  SendSCPI("TRIGGER:SOURCE BUS"); //set trigger source to bus

  /* configure the list*/
   SendSCPI("LIST:MODE CONT"); //set the list mode to
continuous

   ErrorStatus = viPrintf(PWS4000,"LIST:COUNT
%d\n",List_count); //set the list count
  CheckError("Unable to set list count");

  ErrorStatus = viPrintf(PWS4000,"LIST:STEP %d\n",List_step);
//set the list step
  CheckError("Unable to set list step");

  /*set the list voltage, current and time */
  for(i = 1; i<= List_step; i+=1)
  {
```

```
  ErrorStatus = viPrintf(PWS4000,"LIST:CURRent %d, %f\n", i,
List_Curr[i-1]); //set the step current
  CheckError("Unable to set list current");

   ErrorStatus = viPrintf(PWS4000,"LIST:VOLTage %d, %f\n",
i, List_Volt[i-1]); //set the step voltage
  CheckError("Unable to set list voltage");

   ErrorStatus = viPrintf(PWS4000,"LIST:WIDTH %d, %f\n", i,
List_Time[i-1]); //set the output voltage
  CheckError("Unable to set list time");

 }

 SendSCPI("LIST:SAVE 1"); // save the list

 SendSCPI("FUNCTION:MODE LIST"); // set the power supply in
list mode

 SendSCPI("TRIGGER"); // trigger the list

 printf("List is running\n");

 /*Check the state of bit 5 of the Operation Condition
Register to determine if list is still running.  */
 do
 {
 delay(500);

    ErrorStatus = viPrintf(PWS4000,
"STATus:OPERation:CONDition?\n");

  CheckError("Unable to write the device");

  ErrorStatus = viScanf(PWS4000,"%d",&OPERreg);

  } while (OPERreg & 0x20);

  printf("List finished\n");

 ClosePort();

  while(1); //return 0;
}
void OpenPort()
{

  //Open communication session with the power supply
  ErrorStatus = viOpenDefaultRM(& defaultRM);
    ErrorStatus =viOpen(defaultRM,
"USB0::0X0699::0X0391::006002206573001001::INSTR",0,0,&PWS4000);
  CheckError("Unable to open the port");
  SendSCPI("SYSTEM:REMOTE");

}
```

```
void SendSCPI(char* pString)
{

 char* pdest;

 strcpy(commandString,pString);
 strcat(commandString, "\n");
 ErrorStatus = viPrintf(PWS4000, commandString);
 CheckError("Can't Write to Power Supply");

 pdest = strchr(commandString, '?'); //Search for query
command
 if (pdest != NULL)
    {
    ErrorStatus = viBufRead(PWS4000, (ViBuf)ReadBuffer,
sizeof(ReadBuffer), VI_NULL);
   CheckError("Can't read from driver");
    strcpy(pString, ReadBuffer);
    }
}
void ClosePort()
{
  viClose(PWS4000);
  viClose(defaultRM);
}

void CheckError(char* pMessage)
  {

  if(ErrorStatus != VI_SUCCESS)
  {
  printf("\n %s",pMessage);
  ClosePort();
  exit(0);
  }

}

void delay(clock_t wait)
  {

  clock_t goal;
  goal = wait + clock();
  while(goal > clock());
  }
```

Appendix C: Default Setup

The following table lists the settings that are restored when you return the power supply to default settings.

Table C-1: Default settings

Menu or system	Default setting
VOLT:PROT	MAX
VOLT:PROT:STAT	OFF
OUTP	OFF
VOLT	1 V
VOLT:RANG	MAX
CURR	0.1 A
OUTP:TIM	60
OUTP:TIM:STAT	OFF

Index

T

W