Programming Guide
HP-IB DC Power Supplies
Series 664xA, 665xA, 667xA
and 668xA
HP Part No. 5960–5597

HEWLETT®
PACKARD

Microfiche Part No. 5960–5598
Printed in USA November 1993
SAFETY GUIDELINES

The beginning of the power supply Operating Manual has a Safety Summary page. Be sure you are familiar with the information on that page before programming the power supply for operation from a controller.

| Warning | ENERGY HAZARD. Power supplies with high output currents (such as the Series 668xA) can provide more than 240 VA at more than 2 V. If the output connections touch, severe arcing may occur resulting in burns, ignition or welding of parts. Take proper precautions before remotely programming the output circuits. |

PRINTING HISTORY

The edition and current revision of this guide are indicated below. Reprints of this guide containing minor corrections and updates may have the same printing date. Revised editions are identified by a new printing date. A revised edition incorporates all new or corrected material since the previous printing date. Changes to the guide occurring between revisions are covered by change sheets shipped with the guide.

Edition 1 November, 1993

© Copyright 1993 Hewlett-Packard Company
This document contains proprietary information protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated into another language without the prior consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.
INTRODUCTION

ABOUT THIS GUIDE
This guide provides remote programming information for the following series of HP-IB programmable power supplies:

- HP 664xA
- HP 665xA
- HP 667xA
- HP 668xA

You will find the following information in the rest of this guide:

Chapter 2  Introduction to SCPI messages structure, syntax, and data formats. Examples of SCPI programs.
Chapter 3  Dictionary of SCPI commands. Table of programming parameters.
Chapter 4  Description of the status registers
Chapter 5  Error messages
Appendix A  SCPI conformance information.
Appendix B  Use of the alternate Compatibility programming language.

DOCUMENTATION SUMMARY

User’s Guide
The Operating Guide, shipped with the power supply, has information helpful to programming the power supply and explains the SCPI commands used for remote calibration. Sample calibration and verification programs are included.

External Documents

SCPI References
The following document will assist you with programming in SCPI:

   - Highly recommended for anyone who has not had previous experience programming with SCPI or TMSL.

HP-IB References
For a basic introduction to the HP-IB, see the following:

   - Highly recommended for those not familiar with the IEEE 488.1 and 488.2 standards.

1. To obtain a copy, contact your local HP Sales and Support Office.
The most important HP-IB documents are your controller programming manuals - HP BASIC, HP-IB Command Library for MS DOS, etc. Refer to these for all non-SCPI commands (e.g., Local Lockout).

The following are two formal documents concerning the HP-IB interface:

1. ANSI/IEEE Std. 488.1-1987 IEEE Standard Digital Interface for Programmable Instrumentation. This defines the technical details of the HP-IB interface. While much of the information is beyond the need of most programmers, it can serve to clarify terms used in this guide and in related documents.

2. ANSI/IEEE Std. 488.2-1987 IEEE Standard Codes, Formats, Protocols, and Common Commands. Recommended as a reference only if you intend to do fairly sophisticated programming. It is helpful for finding the precise definitions of certain types of SCPI message formats, data types, or common commands.

2Available from the IEEE (Institute of Electrical and Electronics Engineers), 345 East 47th Street, New York, NY 10017, USA.

PREREQUISITES FOR USING THIS GUIDE

The organization of this guide assumes that you know or can learn the following information:

1. How program in your controller language (HP BASIC, QUICKBASIC, C, etc.).
2. The basics of the HP-IB (IEEE 488).
3. How to program I/O statements for an IEEE 488 bus instrument. From a programming aspect, the power supply is simply a bus instrument.
4. How to format ASCII statements within your I/O programming statements. SCPI commands are nothing more than ASCII data strings incorporated within those I/O statements.
5. The basic operating principles of the power supply as explained in “Chapter 5 - Front Panel Operation” of the Operating Guide.
6. How to set the HP-IB address of the power supply. This cannot be done remotely, but only from the supply’s front panel (see System Considerations in “Chapter 2 - Remote Programming”).
Remote Programming

HP-IB CAPABILITIES OF THE POWER SUPPLY
All power supply functions except for setting the HP-IB address are programmable over the IEEE 488 bus (also known as the Hewlett-Packard Interface Bus or “HP-IB”). The IEEE 488.1 capabilities of the power supply are listed in the Supplemental Characteristics of the Operating Guide. The power supply operates from an HP-IB address that is set from the front panel (see System Considerations at the end of this chapter).

INTRODUCTION TO SCPI

**Important!** Learn the basics of power supply operation (see “Chapter 5 - Front Panel Operation” in the power supply Operating Guide) before using SCPI.

SCPI (Standard Commands for Programmable Instruments) is a programming language for controlling instrument functions over the HP-IB (IEEE 488) instrument bus. SCPI is intended to function with standard HP-IB hardware and conforms to the IEEE Standard Digital Interface for Programmable Instrumentation. SCPI is layered on top of the hardware portion of IEEE 488.2. The same SCPI commands and parameters control the same functions in different classes of instruments. For example, you would use the same DISPLAY command to control the power supply display state and the display state of a SCPI-compatible multimeter.

**Note**
HPSL (Hewlett-Packard System Language) and TMSL (Test and Measurement System Language) were earlier versions of SCPI. If you have programmed in either, then you probably can go directly to “Chapter 3 - Language Dictionary”.

**Conventions**
The following conventions are used throughout this chapter:

**Angle brackets** `< >` Items within angle brackets are parameter abbreviations. For example, `<NR1>` indicates a specific form of numerical data.

**Vertical bar** `|` Vertical bars separate one of two or more alternative parameters. For example, 0|OFF indicates that you may enter either “0” or “OFF” for the required parameter.

**Square Brackets** `[]` Items within square brackets are optional. The representation [SOURce]:CURRent means that SOURCee may be omitted.

**Braces** `{ }` Braces indicate parameters that may be repeated zero or more times. It is used especially for showing arrays. The notation `<A>{<,B>}` shows that “A” is a required parameter, while “B” may be omitted or may be entered one or more times.
Boldface font is used to emphasize syntax in command definitions. TRIGger:DELay <NRF> shows a command syntax.

Computer font is used to show program text within normal text. TRIGger:DELay .5 represents program text.

**SCPI Messages**

There are two types of SCPI messages, program and response.

- **A program message** consists of one or more properly formatted SCPI commands sent from the controller to the power supply. The message, which may be sent at any time, requests the power supply to perform some action.

- **A response message** consists of data in a specific SCPI format sent from the power supply to the controller. The power supply sends the message only when commanded by a special program message called a “query.”

**Types of SCPI Commands**

SCPI has two types of commands, common and subsystem.

**Common Commands**

*Common* commands (see Figure 3-1) generally are not related to specific operation but to controlling overall power supply functions, such as reset, status, and synchronization. All common commands consist of a three-letter mnemonic preceded by an asterisk:

* *RST  *IDN?  *SRE 8

**Subsystem Commands**

Subsystem commands (see Figure 3-2) perform specific power supply functions. They are organized into an inverted tree structure with the “root” at the top. Some are single commands while others are grouped under other subsystems.

**Structure of a SCPI Message**

SCPI messages consist of one or more message units ending in a message terminator. The terminator is not part of the syntax, but implicit in the way your programming language indicates the end of a line (such as a newline or end-of-line character).

**The Message Unit**

The simplest SCPI command is a single message unit consisting of a command header (or keyword) followed by a message terminator.

```
ABOR
VOLT?
```

The message unit may include a parameter after the header. The parameter usually is numeric, but it can be a string:

```
VOLT 20
VOLT MAX
```

**Combining Message Units**

The following command message (see Figure 2-1) is briefly described here, with more details in subsequent paragraphs.

```
VOLT:LEV 1.5;PROT 4.8;:CURR?<NL>
```
The basic parts of the message in Figure 2-1 are:

<table>
<thead>
<tr>
<th>Message Component</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td>VOLT LEV PROT CURR</td>
</tr>
<tr>
<td>Header Separator</td>
<td>The colon in VOLT:LEV</td>
</tr>
<tr>
<td>Data</td>
<td>4.5 4.8</td>
</tr>
<tr>
<td>Data Separator</td>
<td>The space in VOLT 4.5 and in PROT 4.8</td>
</tr>
<tr>
<td>Message Units</td>
<td>VOLT:LEV 4.5 PROT 4.8 CURR?</td>
</tr>
<tr>
<td>Message Unit Separator</td>
<td>The semicolons in VOLT:LEV 4.5; and PROT 4.8;</td>
</tr>
<tr>
<td>Root Specifier</td>
<td>The colon in PROT 4.8::CURR?</td>
</tr>
<tr>
<td>Query Indicator</td>
<td>The question mark in CURR?</td>
</tr>
<tr>
<td>Message Terminator</td>
<td>The &lt;NL&gt; (newline) indicator. Terminators are not part of the SCPI syntax</td>
</tr>
</tbody>
</table>

**Parts of a SCPI Message**

**Headers**

*Headers* (which are sometimes known as "keywords") are instructions recognized by the power supply interface. Headers may be either in the long form or the short form.

**Long Form** The header is completely spelled out, such as **VOLTAGE STATUS DELAY**

**Short Form** The header has only the first three or four letters, such as **VOLT STAT DEL.**

Short form headers are constructed according to the following rules:

- If the header consists of *four or fewer* letters, use all the letters. (**DFI DATA**)
- If the header consists of *five or more* letters and the fourth letter is *not* a vowel (a,e,i,o,u), use the first four letters. (**VOLTage STATus**)
- If the header consists of *five or more* letters and the fourth letter is a vowel (a,e,i,o,u), use the first three letters. (**DElay CLEar**)

You must follow the above rules when entering headers. Creating an arbitrary form, such as **QUEST for QUESTIONABLE**, will result in an error. The SCPI interface is *not* sensitive to case. It will recognize any case mixture, such as **VOLTAGE, Voltage, Volt, volt.**

**Introduction to SCPI**

Remote Programming 2-3
**Note**

Shortform headers result in faster program execution.

---

**Header Convention.** In this manual, headers are emphasized with **boldface** type. The proper short form is shown in upper-case letters, such as **DElay**.

**Header Separator.** If a command has more than one header, you must separate them with a colon (VOLT:PROT   OUTPut:PROTection:CLEar).

**Optional Headers.** The use of some headers is optional. Optional headers are shown in brackets, such as OUTPut[:STATe] ON. However, if you combine two or more message units into a compound message, you may need to enter the optional header. This is explained under “Traversing the Command Tree.”

**Query Indicator**
Following a header with a question mark turns it into a query (VOLT?   VOLT:PROT?).
If a query contains a parameter, place the query indicator at the end of the last header (VOLT:PROT? MAX).

**Message Unit Separator**
When two or more message units are combined into a compound message, separate the units with a semicolon (STATus:OPERation?):QUESTIONable?).

---

**Important**
You can combine message units only at the current path of the command tree (see “Traversing the Command Tree”).

---

**Root Specifier**
When it precedes the first header of a message unit, the colon becomes a “root specifier”. This indicates that the command path is at the root or top node of the command tree. Note the difference between root specifiers and header separators in the following examples:

OUTP:PROT:DEL .1   All colons are header separators
:OUTP:PROT:DEL .1   The first colon is a root specifier
OUTP:PROT:DEL .1;VOLT 12.5 The third colon is a root specifier

**Message Terminator**
A terminator informs SCPI that it has reached the end of a message. Three permitted messages terminators are:

- newline (<NL>), which is ASCII decimal 10 or hex 0A.
- end or identify (<END>)
- both of the above (<NL><END>).

In the examples of this manual, there is an assumed message terminator at the end of each message. If the terminator needs to be shown, it is indicated as <NL> regardless of the actual terminator character.

---

**Traversing the Command Tree**
Figure 2-2 shows a portion of the subsystem command tree (you can see the complete tree in Figure 3-2). Note the location of the **ROOT** node at the top of the tree. The SCPI interface is at this location when:

- the power supply is powered on
- a device clear (DCL) is sent to the power supply

---

2-4 Remote Programming
Figure 2-2. Partial Command Tree

- the interface encounters a message terminator
- the interface encounters a root specifier

Active Header Path
In order to properly traverse the command tree, you must understand the concept of the active header path. When the power supply is turned on (or under any of the other conditions listed above), the active path is at the root. That means the interface is ready to accept any command at the root level, such as TRIGger or STATus in Figure 2-2. Note that you do not have to precede either command with a colon; there is an implied colon in front of every root-level command.

If you enter STATus, the active header path moves one colon to the right. The interface is now ready to accept :OPERATION, :PRESET, or QUESTIONable as the next header. Note that you must include the colon, because it is required between headers.

If you next enter :OPERATION, the active path again moves one colon to the right. The interface is now ready to accept EVENT?, CONDITION?, ENABLE, NTRANSITION, or PTRANSITION as the next header.

If you now enter :ENABLE, you have reached the end of the command string. The active header path remains at :ENABLE. If you wished, you could have entered :ENABLE 18; PTRANSITION 18 and it would be accepted. The entire message would be STATus:OPERATION:ENABLE 18; PTRANSITION 18. The message terminator after PTRANSITION 18 returns the path to the root.

The Effect of Optional Headers
If a command includes optional headers, the interface assumes they are there. For example, if you enter STATus:OPERATION?, the interface recognizes it as STATus:OPERATION:EVENT? (see Figure 2-2). This returns the active path to the root (:STATus). But if you enter STATus:OPERATION:EVENT?, then the active path remains at :EVENT. This allows you to send STATus:OPERATION:EVENT?:CONDITION? in one message. If you tried to send STATus:OPERATION?:CONDITION? the command path would send STATus:OPERATION:EVENT? and then return to :STATus instead of to :CONDITION.

The optional header SOURCE precedes the current, digital, and voltage subsystems (see Figure 3-2). This effectively makes :CURRENT, :DIGITAL, and :VOLTAGE root-level commands.
Moving Among Subsystems
In order to combine commands from different subsystems, you need to be able to restore the active path to the root. You do this with the root specifier (:). For example, you could clear the output protection and check the status of the Operation Condition register as follows (see Figure 3-2):

```
OUTPUT:PROTECTION:CLEAR
STATUS:OPERATION:CONDITION?
```

By using the root specifier, you could do the same thing in one message:

```
OUTPUT:PROTECTION:CLEAR;:STATUS:OPERATION:CONDITION?
```

Note
The SCPI parser traverses the command tree as described in Appendix A of the IEEE 488.2 standard. The “Enhanced Tree Walking Implementation” given in that appendix is not implemented in the power supply.

The following message shows how to combine commands from different subsystems as well as within the same subsystem (see Figure 3-2):

```
VOLTAGE:LEVEL 7;:PROTECTION 8;:CURRENT:LEVEL 150;PROTECTION ON
```

Note the use of the optional header LEVEL to maintain the correct path within the voltage and current subsystems and the use of the root specifier to move between subsystems.

Including Common Commands
You can combine common commands with system commands in the same message. Treat the common command as a message unit by separating it with the message unit separator. Common commands do not affect the active header path; you may insert them anywhere in the message.

```
VOLT:TRIG 7.5;INIT;:*TRG
OUTP OFF;:*RCL 2;OUTP ON
```

SCPI Queries
Observe the following precautions with queries:

- Remember to set up the proper number of variables for the returned data.
- Set the program to read back all the results of a query before sending another command to the power supply. Otherwise, a Query Interrupted error will occur and the unreturned data will be lost.

Value Coupling
Value coupling results when a command directed to send one parameter also changes the value of a second parameter. There is no direct coupling among any power supply SCPI commands. However, be aware that until they are programmed, uninitialized trigger levels will assume their corresponding immediate levels. For example, if a power supply is powered up and VOLT:LEV is programmed to 6, then VOLT:LEV:TRIG will also be 6 until you program it to another value. Once you program VOLT:LEV:TRIG to another value, it will remain at that value regardless of how you subsequently reprogram VOLT:LEVEL.

SCPI Data Formats
All data programmed to or returned from the power supply is ASCII. The data may be numerical or character string.
Numerical Data

Table 2-1 and Table 2-2 summarize the numerical formats.

Table 2-1. Numerical Data Formats

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data Form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Talking Formats</strong></td>
</tr>
<tr>
<td>&lt;NR1&gt;</td>
<td>Digits with an implied decimal point assumed at the right of the least-significant digit. Examples: 273 0273</td>
</tr>
<tr>
<td>&lt;NR2&gt;</td>
<td>Digits with an explicit decimal point. Example: 273.0273</td>
</tr>
<tr>
<td>&lt;NR3&gt;</td>
<td>Digits with an explicit decimal point and an exponent. Example: 2.73E+2 273.0E-2</td>
</tr>
<tr>
<td></td>
<td><strong>Listening Formats</strong></td>
</tr>
<tr>
<td>&lt;NRF&gt;</td>
<td>Extended format that includes &lt;NR1&gt;, &lt;NR2&gt; and &lt;NR3&gt;. Examples: 273 273.273E2</td>
</tr>
<tr>
<td>&lt;NRF+&gt;</td>
<td>Expanded decimal format that includes &lt;NRF&gt;, MIN, and MAX. Examples: 273 273.273E2 MAX. MIN and MAX are the minimum and maximum limit values that are implicit in the range specification for the parameter.</td>
</tr>
</tbody>
</table>

Table 2-2. Suffixes and Multipliers

<table>
<thead>
<tr>
<th>Class</th>
<th>Suffix</th>
<th>Unit</th>
<th>Unit with Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>A</td>
<td>Ampere</td>
<td>MA (milliampere)</td>
</tr>
<tr>
<td>Amplitude</td>
<td>V</td>
<td>Volt</td>
<td>MV (millivolt)</td>
</tr>
<tr>
<td>Time</td>
<td>S</td>
<td>second</td>
<td>MS (millisecond)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Common Multipliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1E3</td>
</tr>
<tr>
<td>1E-3</td>
</tr>
<tr>
<td>1E-6</td>
</tr>
</tbody>
</table>

Boolean Data

Either form {1/3} or {ON/OFF} may be sent with commands. Queries always return 1 or 0.

```
OUTPut OFF
CURRent:PROTection 1
```

Character Data

For query statements, character strings may be returned in either of the forms shown in Table 2-3, depending on the length of the returned string.

Table 2-3. Character Data Formats

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;CRD&gt;</td>
<td>Character Response Data. Permits the return of character strings.</td>
</tr>
<tr>
<td>&lt;AARD&gt;</td>
<td>Arbitrary ASCII Response Data. Permits the return of undelimited 7-bit ASCII. This data type has an implied message terminator.</td>
</tr>
</tbody>
</table>

Note: The IEEE 488.2 format for a string parameter requires that the string be enclosed within either single (’ ') or double (" ") quotes. Be certain that your program statements comply with this requirement.
EXAMPLES
The examples given here are generic, without regard to the programming language or type of HP-IB interface. Because SCPI commands are sent as ASCII output strings within the programming language statements, the SCPI syntax is independent of both programming language and interface.

Note The examples are followed by sample program code written for three popular types of BASIC-controlled HP-IB interfaces.

Controlling Output

Important The power supply responds simultaneously to both digital and analog programming inputs. If it is receiving an input over the HP-IB and a corresponding input from the front panel (and/or from the analog programming port), the power supply output will be the algebraic sum of the inputs.

Programming Voltage and Current
The following statements program both voltage and current and return the actual output from the sense terminals:

\[ \text{OUTP OFF} \quad \text{Disable the output} \]
\[ \text{VOLT 4.5; CURR 255} \quad \text{Program the voltage and current} \]
\[ \text{VOLT?; CURR?} \quad \text{Read back the programmed levels} \]
\[ \text{OUTP ON} \quad \text{Enable the output} \]
\[ \text{MEAS: VOLT?; MEAS: CURR?} \quad \text{Read back the outputs from the sense terminals} \]

Programming Protection Circuits
This example programs the voltage and current, programs an overvoltage protection value, and turns on the overcurrent protection. It then reads back all the programmed values.

\[ \text{VOLT: LEV 4.5; PROT 4.75} \quad \text{Program the voltage and overvoltage protection} \]
\[ \text{CURR: LEV 255; PROT: STAT ON} \quad \text{Program the current and overcurrent protection} \]
\[ \text{VOLT: LEV?; PROT?; CURR: LEV?; PROT: STAT?} \quad \text{Read back the programmed values} \]

Note the required use of the optional LEVel header in the above example (see “The Effect of Optional Headers”, given previously).

Changing Outputs by Trigger
If you do not program pending triggered levels, they default to the programmed (immediate) output levels. The following statements shows some basic trigger commands.

\[ \text{OUTP OFF} \quad \text{Disable the output} \]
\[ \text{VOLT: LEV IMM 2.2; TRIG 2.5} \quad \text{Program the (immediate) voltage level to 2.2 V and the pending triggered level to 2.5 V.} \]
\[ \text{CURR: LEV IMM 150; TRIG 250} \quad \text{Program the (immediate) current level to 150 A and the pending triggered level to 250 A.} \]
VOLT:LEV:IMM?;TRIG?;CURR:LEV:IMM?;TRIG? Check all the programmed values.
OUTP ON Enable the output.
MEAS:VOLT?;CURR? Read back the immediate levels from the sense terminals.
INIT;TRIG Arm the trigger circuit and send a single trigger.
INIT:*TRG Same as above, except using a common command.
MEAS:VOLT?;CURR? Read back the triggered levels from the sense terminals.

If you need to send two or more triggers, program the trigger circuit for continuous arming.

OUTP OFF Disable the output.
VOLT:LEV:IMM 5.0;TRIG 2.5 Program the (immediate) voltage level to 5 V and the pending triggered level to 2.5 V.
INIT:CONT ON Program the trigger circuit for continuous arming.
OUTP ON Enable the output to 5 V.
TRIG Trigger the output voltage to 2.5 V.
VOLT:TRIG 5.;TRIG Set the pending trigger level to 5 V and trigger the output voltage back to 5 V.
INT:CONT OFF Remove the continuous trigger arming.

Saving and Recalling States
You can remotely save and recall operating states. See *SAV and *RCL in “Chapter 3 - Language Dictionary” for the parameters that are saved and recalled.

Note
When you turn the power supply on, it automatically retrieves the state stored in location 0. When a power supply is delivered, this location contains the factory defaults (see *RST in “Chapter 3 - Language Dictionary”).

OUTP OFF;VOLT:LEV 6.5; PROT 6.8 Program a desired operating state
CURR:LEV 335; PROT:STAT ON
*SAV 2 Save this state to location 2
*RCL 2 (Later) recall this same state

Writing to the Display
You can include messages to the front panel LCD in your programs. The description of DISP:TEXT in “Chapter 3 - Language Dictionary” shows the number and types of permitted display characters. In order to write to the display, you must first change it to text mode as shown in the following example:
DIS:MODE TEXT Switch display to text mode
RECALLED 2 Write “Recalled 2” to the display
DIS:MODE NORM Return display to its normal mode

Examples
Programming Status
You can use status programming to make your program react to events within the power supply. "Chapter 4 - Status Reporting" explains the functions and bit configurations of all status registers. Refer to Figure 4-1 in that chapter while examining the examples given here.

Detecting Events via SRQ
Usually you will want the power supply to generate interrupts (assert SRQ) upon particular events. For this you must selectively enable the appropriate status register bits. The following examples allow the power supply to assert SRQ under selected conditions.

1. STAT:OPER:ENAB 1280;PTR 1280;*SRE 128  Assert SRQ when the supply switches between CV and CC modes
2. STAT:OPER:ENAB 1;PTR 1;NTR 1;*SRE 128  Assert SRQ when the supply enters or leaves calibration mode
3. STAT:QUES 3;PTR 3;*SRE 128  Assert SRQ when the supply goes into overvoltage or overcurrent condition
4. STAT:OPER:ENAB 1280;PTR 1280;STAT:QUES 3;PTR 3;*SRE 136  Assert SRQ under any event occurring in 1. or 3., above

Reading Specific Registers
You can exercise program control without interrupts by reading specific registers.

STAT:OPER:1280;EVEN?  Enable only the CV and CC events and read their status
STAT:OPER:ENAB 1313;PTR 1313;EVEN?  Enable all conditions of the Operation Status register and read any events
STAT:OPER:ENAB?;EVENT?;:STAT:QUES:ENAB?;EVEN?;:*ESE?;*ESR?  Read which events are active and which events are enabled in the Operation, Questionable, and Standard Event status registers

Note
The last query string can be handled without difficulty. However, you should request too many queries, the system may return a "Query DEADLOCKED" error (-430). In that case, break the long string into smaller parts.

Programming the Digital I/O Port
Digital control ports 1 and 2 are TTL outputs that can be programmed either high or low. Control port 3 can be programmed to be either a TTL input or a TTL output. Send a decimal parameter that translates into the desired straight binary code for these ports. (See DIG:DATA[:VAL] in "Chapter 3 - Language Dictionary" for the port bit configurations.)

DIG:DATA 3  Set ports 1 and 2 high and make 3 another output port
DIG:DATA 7  Set ports 1 and 2 high and make 3 an input port
DIG:DATA?  Read back the present port configuration
SYSTEM CONSIDERATIONS

The remainder of this chapter addresses some system issues concerning programming. These are power supply addressing and the use of the following types of HP-IB system interfaces:

1. HP Vectra PC controller with IHP 8235A HP-IB Interface Command Library
2. IBM PC controller with National Instruments GPIB-PCII Interface/Handler
3. HP controller with HP BASIC Language System

The HP-IB Address

The power supply address cannot be set remotely; it must be set from the front panel. Once the address is set, you can assign it inside programs.

Setting the HP-IB Address

Figure 4-6 in the power supply Operating Guide shows the ways the power supply can be connected to the HP-IB bus. You can set up the HP-IB address in one of three ways:

1. As a stand-alone supply (the only supply at the address). It has a primary address in the range of 0 to 30. For example:
   5 or 7
2. As the direct supply in a serial link. It is the only supply connected directly to the HP-IB bus. The primary address is unique and can be from 0 to 30. It is entered as an integer followed by a decimal separator. The secondary address always is 0, which may be added after the primary address. If the secondary address is omitted, it is assumed to be 0. For example:
   5.0 or 7.
3. As a linked supply in serial link. It gets its primary address from the direct supply. It has a unique secondary address that can be from 1 to 15. It is entered as an integer preceded by a decimal separator. For example:
   .1 or .12

When you enter a secondary address, leading zeros between the decimal separator and the first digit are ignored. For example, .1, .01, and .001 are accepted as secondary address 1 and displayed as 0.01. Zeros following a digit are not ignored. Thus, .10 and .010 are both accepted as secondary address 10 and displayed as 0.10.

Changing the Power Supply HP-IB Address

Use the Address key and numerical keypad for entering addresses. The power supply is shipped with a 5 stand-alone address as the default. The general procedure for setting an address is:

<table>
<thead>
<tr>
<th>Action</th>
<th>Display Shows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Press Address</td>
<td>Current address</td>
</tr>
<tr>
<td>Press new address keys</td>
<td>New address replaces numbers on the display</td>
</tr>
<tr>
<td>Press Enter</td>
<td>Display returns to meter mode</td>
</tr>
</tbody>
</table>

If you try to enter a forbidden number, ADDR ERROR is displayed.

The following examples show how to set addresses:

- To set stand-alone primary address 6, press Address 6 Enter
- To set direct supply primary address 6, press Address 6.0 Enter
- To set linked secondary address 1, press Address 1.0 Enter
- To set linked secondary address 12, press Address 1.2 Enter
Note

The power supply display will reset (recall the state in location 0) whenever you change between the following types of HP-IB addresses:

- a stand-alone primary address and a direct primary address.
- a direct primary address and a secondary address.

Assigning the HP-IB Address in Programs

The following examples assume that the HP-IB select code is 7, the power supply is 6, and that the power supply address will be assigned to the variable @PS.

1000 !Stand-alone address. The power supply will respond if it is set to 6
1010 PS=706 !Statement for HP82335A Interface
1010 ASSIGN @PS TO 706 ! Statement for HP BASIC Interface
1020 !Direct address. The power supply will respond if it is set to 6 or 6.0
1030 PS=70600 !Statement for HP82335A Interface
1030 ASSIGN @PS TO 70600 ! Statement for HP BASIC Interface
1040 !Linked address 1. The power supply will respond if it is set to address .1 and is serially connected to a supply at direct address 6.0
1050 PS=706.01 !HP82335A Interface
1090 ASSIGN @PS TO 706.01 !HP BASIC Interface

For systems using the National Instruments DOS driver, the address is specified in the software configuration program (IBCONFIG.EXE) and assigned a symbolic name. The address then is referenced only by this name within the application program (see the National Instruments GP-IB documentation).

DOS Drivers

Types of Drivers

The HP 82335A and National Instruments GP-IB are two popular DOS drivers. Each is briefly described here. See the software documentation supplied with the driver for more details.

HP 82335A Driver. For GW-BASIC programming, the HP-IB library is implemented as a series of subroutine calls. To access these subroutines, your application program must include the header file SETUP.BAS, which is part of the DOS driver software.

SETUP.BAS starts at program line 5 and can run up to line 999. Your application programs must begin at line 1000. SETUP.BAS has built-in error checking routines that provide a method to check for HP-IB errors during program execution. You can use the error-trapping code in these routines or write your own code using the same variables as used by SETUP.BAS.

National Instruments GP-IB Driver. Your program must include the National Instruments header file DECL.BAS. This contains the initialization code for the interface. Prior to running any applications programs, you must set up the interface with the configuration program (IBCONF.EXE).
Your application program will not include the power supply symbolic name and HP-IB address. These must be specified during configuration (when you run IBCONF.EXE). Note that the primary address range is from 0 to 30 but any secondary address must be specified in the address range of 96 to 126. The power supply expects a message termination on EOI or line feed, so set EOI w/last byte of Write. It is also recommended that you set Disable Auto Serial Polling.

All function calls return the status word IBSTA\%, which contains a bit (ERR) that is set if the call results in an error. When ERR is set, an appropriate code is placed in variable IBERR\%. Be sure to check IBSTA\% after every function call. If it is not equal to zero, branch to an error handler that reads IBERR\% to extract the specific error.

**Error Handling**
If there is no error-handling code in your program, undetected errors can cause unpredictable results. This includes “hanging up” the controller and forcing you to reset the system. Both of the above DOS drivers have routines for detecting program execution errors.

---

**Important**
Use error detection after every call to a subroutine.

---

**HP BASIC Controllers**
The HP BASIC Programming Language provides access to HP-IB functions at the operating system level. This makes it unnecessary to have the header files required in front of DOS applications programs. Also, you do not have to be concerned about controller “hangups” as long as your program includes a timeout statement. Because the power supply can be programmed to generate SRQ on errors, your program can use an SRQ service routine for decoding detected errors. The detectable errors are listed in Table 5-1 of “Chapter 5 - Error Messages”.

**Sample Program Code**
The following programs are intended only to show how some of the same power supply functions can be programmed to each of the three previously mentioned HP-IB interfaces. The first two are for the DOS interfaces and the third for the HP BASIC interface.
SAMPLE FOR POWER SUPPLY AT STAND-ALONE ADDRESS 6. SEQUENCE SETS UP CV MODE OPERATION,
FORCES SUPPLY TO SWITCH TO CC MODE, AND DETECTS AND REPORTS MODE CHANGE.

*******************************************************************************
HP Vectra PC Controller Using HP 82335A Interface
*******************************************************************************
5   ' <----------------- Merge SETUP.BAS here ----------------------------->
1000 MAXELEMENTS=2 :ACTUAL ELEMENTS=0 :MAX.LENGTH=80 :ACT.LENGTH=0
1005 DIM OUTPUTS(2) :CODES$=SPACE$(40)
1010 ISC=7 :PS=706
1015 ,
1020 'Set up the Power Supply Interface for DOS driver
1025 CALL IORESET (ISC)   'Reset the interface
1030 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1035 TIMEOUT=3
1040 CALL IOT:MEOUT (ISC, TIMEOUT)   'Set timeout to 3 seconds
1045 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1050 CALL IOCLEAR (ISC)   'Clear the interface
1055 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1060 CALL IOMODE (ISC)   'Set Power Supply to remote mode
1065 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1070 ,
1075 'Program power supply to CV mode with following voltage and current
1080 CODES$ = "VOLTAGE 7.8;CURRENT 480" :GOSUB 2000
1085 ,
1090 'Query power supply outputs & print to screen
1095 CODES$ = "MEASURE:VOLTAGE;CURRENT" :GOSUB 2000 :GOSUB 3000
1100 VOUT = OUTPUTS(1)
1105 IGOT = OUTPUTS(2)
1110 PRINT "The output levels are "VOUT" Volts and "IGOT" Amps"
1115 ,
1120 'Program triggered current level to value insufficient to maintain
1125 'supply within its CV operating characteristic
1130 CODES$ = "CURR:TRIG 60" :GOSUB 2000
1135 ,
1140 'Set operation status mask to detect mode change from CV to CC
1145 CODES$ = "STAT:OPER:ENAB 1024;PTR 1024" :GOSUB 2000
1150 ,
1155 'Enable Status Byte OPER summary bit
1160 CODES$ = "=SRE 128" :GOSUB 2000
1165 ,
1170 'Arm trigger circuit and send trigger to power supply
1175 CODES$ = "INITIA" :GOSUB 2000
1180 ,
1185 'Wait for supply to respond to trigger
1190 FOR I= 1 to 100 :NEXT I
1195 ,
1200 'Poll for interrupt caused by change to CC mode and print to screen
1205 CALL IOSPLoll (PS,RESPONSE)
1210 IF (RESPONSE AND 128)<=128 THEN GOTO 1240   'No OPER event to report
1215 CODES$ = "STATUS:OPER:EVEN?" :GOSUB 2000 'Query status oper register
1220 CALL IOEVENT (PS,EVENT)   'Read back event bit
1225 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
1230 IF (EVENT AND 1024) = 1024 THEN PRINT "Supply switched to CC mode."
'Clear the status circuit
CODES$ = "*CLS" :GOSUB 2000
FOR I = 1 TO 100 :NEXT I  'Wait for supply to clear

'Disable output and save present state in location 2
CODES$ = "OUTPUT OFF;*SAV 2" :GOSUB 2000
END

'Send command to power supply
LENGTH = LEN(CODES$)
CALL IODOUTPUTS (PS,CODES$,LENGTH)  'Send command to interface
IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR  'SETUP.BAS error trap
RETURN

'Get data from power supply
CALL IDENTERA (PS,OUTPUTS(1),MAX.ELEMENTS,ACTUAL.ELEMENTS)
IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
RETURN

****************************
IBM Controller Using National Interface
****************************

'Power Supply Variable = PS% ; Stand-Alone Address = 706
CODES$=SPACE$(50);MODE$=SPACE$(5);EVENT$=SPACE$(20)
PS$=SPACE$(60);OUTPUT$=SPACE$(40);BDNAME$="PS%"
DIM OUTPUT(2)

'Set up power supply interface for DOS driver
CALL IBFIND(BDNAME$,.PS%)
IF PS%<0 THEN PRINT "IBFIND Failed."
CALL IBICLA(PS%)

'Program power supply to CV mode with following voltage and current
CODES$ = "VOLTAGE 7.8;CURRENT 480" :GOSUB 2000

'Query power supply outputs and print to screen
CODES$ = "MEASURE:VOLTAGE?;CURRENT?" :GOSUB 2000 :GOSUB 3000
VOUT = OUTPUT(1)
IOUT = OUTPUT(2)
PRINT"The programmed levels are "VOUT" Volts and "IOUT" Amps"

'Program triggered current level to value insufficient to maintain
'supply within its CV operating characteristic
CODES$ = "CURR:TRIG 50" :GOSUB 2000

'Set operation status mask to detect mode change from CV to CC
CODES$ = "STAT:OPER:ENAB 1024;PTR 1024" :GOSUB 2000

'Enable Status Byte OPER summary bit
CODES$ = "*SRE 128" :GOSUB 2000

'Arm trigger circuit and send trigger to power supply
CODES$ = "INITIATE;TRIGGER" :GOSUB 2000
'Wait for supply to respond to trigger
FOR I = 1 to 100 : NEXT I
'Poll for interrupt caused by change to CC mode and print to screen
SPOL% = 0
CALL IBRSP(PS%, SPOL%)
IF (SPOL% AND 128) = 128 THEN POLL = 1 'Set interrupt flag on OPER bit
IF POLL <> 1 THEN GOTO 1230 'No interrupt to service
"CODES$ = "STAT:OPER:EVEN?" : GOSUB 2000 'Query status oper register
CALL IBRD(PS%, DEVENT$) 'Read back event bit
IF IBSTAX < 0 THEN GOTO 2100
OEVENT = VAL(DEVENT$)
IF (DEVENT AND 1024) = 1024 THEN PRINT "Supply switched to CC mode."
'
'Clear status circuit
CODES$ = "CLS" : GOSUB 2000
FOR I = 1 TO 50 : NEXT I 'Wait for supply to clear
'
'Disable output and save present state to location 2
CODES$ = "OUTPUT OFF; SAV 2" : GOSUB 2000
END
'Send command to power supply
CALL IBWRP(PS%, CODES$)
IF IBSTAX < 0 THEN GOTO 2100 'Error detected
RETURN
'Disable output and save present state to location 2
CODES$ = "OUTPUT OFF; SAV 2" : GOSUB 2000
END
'Send command to power supply
CALL IBWRP(PS%, CODES$)
IF IBSTAX < 0 THEN GOTO 2100 'Error detected
RETURN
'Send command to power supply
CALL IBWRP(PS%, CODES$)
IF IBSTAX < 0 THEN GOTO 2100 'Error detected
RETURN
'Error detection routine
PRINT "GPIB error. IBSTAX% = &H"; HEX$(IBSTAX%)
PRINT " IBERX% = "; IBERX% in line "; ERL
STOP
'Get data from power supply
CALL IBRD(PS%, OUTPUT$)
IF IBSTAX < 0 THEN GOTO 2100
I = 1 'Parse data string
X = 1
C = INSTR(I, OUTPUT$, ";")
WHILE C <> -1
D$ = MID$(OUTPUT$, I, C - I)
OUTPUT(X) = VAL(D$) 'Get values
I = C + 1
C = INSTR(I, OUTPUT$, ";")
X = X + 1
WEND
3065  DS=RIGHT$(OUTPUT$ LEN(OUTPUT$)-(I-1))
3070  OUTPUT(X)=VAL(DS)
3075  OUTPUT$=SPACE$(40)  'Clear string
3080  RETURN

*******************************************************************************
  Controller Using HP BASIC
*******************************************************************************
1000  !Power supply at stand-alone address = 706
1005  OPTION BASE 1
1010  DIM Codes$(80),Response$(80),Mode$(32)
1015  
1020  !Program power supply to CV mode with following voltage and current
1025  OUTPUT 706:"VOLTAGE 7.8;CURRENT 480"
1030  !Query power supply outputs and print to screen
1035  !Query output levels
1040  OUTPUT 706:"MEASURE:VOLTAGE?;CURRENT?"
1045  ENTER 706;Vout,Iout
1050  PRINT "The output levels are ",Vout," Volts and ",Iout," Amps"
1055  !Program current triggered level to a value insufficient to maintain
1060  !supply within its CV operating characteristic
1065  OUTPUT 706:"CURR:TRIG 50"
1070  !Set operation status mask to detect mode change from CV to CC
1075  OUTPUT 706:"STAT:OPER:ENAB 1280;PTR 1280"
1080  !Enable Status Byte OPER summary bit
1085  OUTPUT 706:"*SRE 128"
1090  !Arm trigger circuit and send trigger to power supply
1095  OUTPUT 706:"INITIATE;TRIGGER"
1100  !Poll for interrupt caused by change to CC mode and print to screen
1105  Response=SPOLL(706)
1110  IF NOT BIT (Response,7) THEN GOTO 1130   !No OPER event to report
1115  OUTPUT 706:"STAT:OPER:EVEN?"
1120  !Query status operation register
1125  ENTER 706;Event
1130  !Read back event bit
1135  IF BIT(Event,10) THEN PRINT "Supply switched to CC mode."
1140  !Clear status
1145  OUTPUT 706:"*CLS"
1150  !Disable output and save present state in location 2
1155  OUTPUT 706:"OUTPUT OFF;*SAV 2"
1160  END

Programming Some Power Supply Functions (continued)
Language Dictionary

INTRODUCTION
This section gives the syntax and parameters for all the IEEE 488.2 SCPI commands and the Common commands used by the power supply. It is assumed that you are familiar with the material in “Chapter 2 - Remote Programming”. That chapter explains the terms, symbols, and syntactical structures used here and gives an introduction to programming. You should also be familiar with “Chapter 5 - Front Panel Operation” (in the Operating Guide) in order to understand how the power supply functions.

The programming examples are simple applications of SCPI commands. Since SCPI syntax remains the same for all programming languages, the examples are generic.

Syntax definitions use the long form, but only short form headers (or “keywords”) appear in the examples. If you have any concern that the meaning of a header in your program listing will not be obvious at some later time, then use the long form to help make your program self-documenting.

Parameters
Most commands require a parameter and all queries will return a parameter. The range for a parameter may vary according to the model of power supply. Parameters for all current models are listed in Table 3-1 at the end of this chapter.

Related Commands
Where appropriate, related commands or queries are included. These are listed either because they are directly related by function or because reading about them will clarify or enhance your understanding of the original command or query.

Order of Presentation
The dictionary is organized as follows:

- IEEE 488.2 common commands, in alphabetical order
- Subsystem commands

Common Commands
Common commands begin with an * and consist of three letters (command) or three letters and a ? (query). Common commands are defined by the IEEE 488.2 standard to perform some common interface functions. The power supply responds to the 13 required common commands that control status reporting, synchronization, and internal operations. The power supply also responds to five optional common commands controlling triggers, power-on conditions, and stored operating parameters.
Subsystem Commands
Subsystem commands are specific to power supply functions. They can be a single command or a group of commands. The groups are comprised of commands that extend one or more levels below the root. The description of subsystem commands follows the listing of the common commands.

DESCRIPTION OF COMMON COMMANDS
Figure 3-1 shows the common commands and queries. These commands are listed alphabetically in the dictionary. If a command has a corresponding query that simply returns the data or status specified by the command, then both command and query are included under the explanation for the command. If a query does not have a corresponding command or is functionally different from the command, then the query is listed separately. The description of each common command or query specifies any status registers affected. In order to make use of this information, you must refer to “Chapter 4 - Status Reporting”, which explains how to read specific register bits and use the information that they return.

Figure 3-1. Common Commands Syntax Diagram
**CLS**

**Meaning and Type**
*Clear Status*    Device Status

**Description**
This command causes the following actions (see “Chapter 4 - Status Reporting” for descriptions of all registers):

- Clears the following registers:
  - Standard Event Status
  - Operation Status Event
  - Questionable Status Event
  - Status Byte
- Clears the Error Queue
- If **CLS** immediately follows a program message terminator (<NL>), then the output queue and the MAV bit are also cleared.

**Command Syntax**  **CLS**
**Parameters** (None)
**Query Syntax** (None)

---

**ESE**

**Meaning and Type**
*Event Status Enable*    Device Status

**Description**
This command programs the Standard Event Status Enable register bits. The programming determines which events of the Standard Event Status Event register (see **ESR**) are allowed to set the ESB (Event Summary Bit) of the Status Byte register. A “1” in the bit position enables the corresponding event. All of the enabled events of the Standard Event Status Event Register are logically ORed to cause the Event Summary Bit (ESB) of the Status Byte Register to be set. See “Chapter 4 - Status Reporting” for descriptions of all three registers.

**Bit Configuration of Standard Event Status Enable Register**

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Name</td>
<td>PON</td>
<td>0</td>
<td>CME</td>
<td>EXE</td>
<td>DDE</td>
<td>QYE</td>
<td>0</td>
<td>OPC</td>
</tr>
<tr>
<td>Bit Weight</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

CME = Command error; DDE = Device-dependent error; EXE = Execution error; OPC = Operation complete; PON = Power-on; QYE = Query error
**ESE**

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>*ESE &lt;NR1&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>0 to 255</td>
</tr>
<tr>
<td>Power On Value</td>
<td>(See <em>PSC</em>)</td>
</tr>
<tr>
<td>Suffix</td>
<td>(None)</td>
</tr>
<tr>
<td>Example</td>
<td>*ESE 129</td>
</tr>
<tr>
<td>Query Syntax</td>
<td>*ESE?</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;NR1&gt; (Register value)</td>
</tr>
<tr>
<td>Related Commands</td>
<td>*ESR?  *PSC  *STB?</td>
</tr>
</tbody>
</table>

**Caution**

If PSC is programmed to 0, then the *ESE command causes a write cycle to nonvolatile memory. The nonvolatile memory has a finite maximum number of write cycles (see Supplemental Characteristics in Chapter 1 of the power supply Operating Guide). Programs that repeatedly cause write cycles to nonvolatile memory can eventually exceed the maximum number of write cycles and may cause the memory to fail.

---

**ESR?**

**Meaning and Type**

_Event Status Register_ Device Status

**Description**

This query reads the Standard Event Status Event register. Reading the register clears it. The bit configuration of this register is the same as the Standard Event Status Enable register (*ESE). See “Chapter 4 - Status Reporting” for a detailed explanation of this register.

<table>
<thead>
<tr>
<th>Query Syntax</th>
<th>*ESR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>(None)</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;NR1&gt; (Register binary value)</td>
</tr>
<tr>
<td>Related Commands</td>
<td>*CLS  *ESE  *ESE?  *OPC</td>
</tr>
</tbody>
</table>

---

**IDN?**

**Identification Query**

**Meaning and Type**

_Identification_ System Interface

**Description**

This query requests the power supply to identify itself. It returns a string composed of four fields separated by commas.

<table>
<thead>
<tr>
<th>Query Syntax</th>
<th>*IDN?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned Parameters</td>
<td>&lt;AARD&gt;</td>
</tr>
</tbody>
</table>

**Information**

_Hewlett-Packard_ Manufacturer

_4xxxx_ 4-digit model number followed by a letter suffix

_1nnnnnnn_ 10-character serial number or 0

_2x.xx_ Revision levels of firmware

<table>
<thead>
<tr>
<th>Example</th>
<th>HEWLETT-PACKARD,6681,0,A.00.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Commands</td>
<td>(None)</td>
</tr>
</tbody>
</table>
*OPC

Meaning and Type
Operation Complete  Device Status

Description
This command causes the interface to set the OPC bit (bit 0) of the Standard Event Status register when the power supply has completed all pending operations. (See *ESE for the bit configuration of the Standard Event Status register.) Pending operations are complete when:

■ all commands sent before *OPC have been executed. This includes overlapped commands. Most commands are sequential and are completed before the next command is executed. Overlapped commands are executed in parallel with other commands. Commands that affect output voltage, current or state, relays, and trigger actions are overlapped with subsequent commands sent to the power supply. The *OPC command provides notification that all overlapped commands have been completed.

■ any change in the output level caused by previous commands has been completed (completion of settling time, relay bounce, etc.)

■ all triggered actions are completed

*OPC does not prevent processing of subsequent commands but Bit 0 will not be set until all pending operations are completed.

Command Syntax  *OPC
Parameters  (None)
Related Commands  *OPC?  *WAI

*OPC?

Meaning and Type
Operation Complete  Device Status

Description
This query causes the interface to place an ASCII “1” in the Output Queue when all pending operations are completed. Pending operations are as defined for the *OPC command. Unlike *OPC, *OPC? prevents processing of all subsequent commands. *OPC? is intended to be used at the end of a command line so that the application program can then monitor the bus for data until it receives the “1” from the power supply Output Queue.

Caution
Do not follow *OPC? with *TRG or HP-IB bus triggers. Such triggers sent after *OPC? will be prevented from executing and will prevent the power supply from accepting further commands. If this occurs, the only programmable way to restore operation is by sending the power supply a HP-IB DCL (Device Clear) command.

Query Syntax  *OPC?
Returned Parameters  <NR1>  ASCII 1 is placed in the Output Queue when the power supply has completed operations.
Related Commands  *OPC  *TRIG  *WAI

Common Commands

Language Dictionary  3-5
*OPT?

Meaning and Type
Option Identification Query

Description
This query requests the power supply to identify any options that are installed. Options are identified by number A 0 indicates no options are installed.

Query Syntax   *OPT?
Returned Parameters   <AARD>

*PSC

Meaning and Type
Power-on Status Clear   Device Initialization

Description
This command controls the automatic clearing at power turn-on of:

- the Service Request Enable register
- the Standard Event Status Enable register

If the command parameter = 1, then the above registers are cleared at power turn on. If the command parameter = 0, then the above registers are not cleared at power turn on but are programmed to their last state prior to power turn on. This is the most common application for *PSC and enables the power supply to generate an SRQ (Request for Service) at power on.

Command Syntax   *PSC <bool>
Parameters   0|1|OFF|ON
Example   *PSC 0   *PSC 1
Query Syntax   *PSC?
Returned Parameters   <NR1>   0|1
Related Commands   *ESE   *SRE

Caution
*PSC causes a write cycle to nonvolatile memory. If *PSC is programmed to 0, then the *ESE and *SRE commands also cause a write cycle to nonvolatile memory. The nonvolatile memory has a finite number of write cycles (see “Table 1-2, Supplementary Characteristics”). Programs that repeatedly write to nonvolatile memory can eventually exceed the maximum number of write cycles and may cause the memory to fail.

*RCL

Meaning and Type
Recall   Device State

Warning
Recalling a previously stored state may place hazardous voltage at the power supply output.

Description
This command restores the power supply to a state that was previously stored in memory with the *SAV command to the specified location. The following states are recalled:
*RCL

CURR:PROT:STAT  OUTP:PROT:DEL  VOLT:[LEV]:[IMM]

Sending *RCL also does the following:

- forces an ABORe command before resetting any parameters (this cancels any uncompleted trigger actions)
- disables the calibration function by setting CAL:STATe to OFF
- sets display functions as follows:
  - DISP:[WIND]:[STATe] to ON
  - DISP:[WIND]:MODE to NORMal
  - DISP:[WIND]:TEXT to ‘ ’
- sets INIT:CONT to OFF
- sets TRIG:SOUR to BUS

At power turnon, the power supply normally is returned to the factory defined turn-on state (see *RST). However, it also may turn on to the state stored in location 0 (see Turn-On Condition under “Chapter 5 - Front Panel Operation” of the power supply Operating Guide).

Command Syntax  *RCL <NRF>
Parameters  0|1|2|3
Example  *RCL 3
Query Syntax  (None)
Related Commands  *PSC  *RST  *SAV

*RST

Meaning and Type
Reset    Device State

Description
This command resets the power supply to a factory-defined state as defined below. *RST also forces an ABORe command.

<table>
<thead>
<tr>
<th>Command</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAL:STAT OFF</td>
<td>OUTP:[STAT] OFF</td>
</tr>
<tr>
<td>CURR:[LEV]:[IMM] *</td>
<td>OUTP:PROT:DEL *</td>
</tr>
<tr>
<td>CURR:[LEV]:TRIG *</td>
<td>OUTP:REL:[STAT] OFF</td>
</tr>
<tr>
<td>CURR:PROT:STAT OFF</td>
<td>OUTP:REL:POL NORM</td>
</tr>
<tr>
<td>DIG:DATA 0</td>
<td>TRIG:SOUR BUS</td>
</tr>
<tr>
<td>DISP:[WIND]:STAT ON</td>
<td>VOLT:[LEV]:[IMM] *</td>
</tr>
<tr>
<td>DISP:[WIND]:MODE NORM</td>
<td>VOLT:[LEV]:[TRIG] *</td>
</tr>
<tr>
<td>DISP:[WIND]:TEXT ' '</td>
<td>VOLT:PROT:[LEV] *</td>
</tr>
<tr>
<td>INIT:CONT OFF</td>
<td></td>
</tr>
</tbody>
</table>

* Model-dependent value. See Table 3-1.

Command Syntax  *RST
Parameters  (None)
Query Syntax  (None)
Related Commands  *PSC  *SAV
**SAV**

**Meaning and Type**

SAVE  Device State

**Description**

This command stores the present state of the power supply to the specified location in memory. Up to four states can be stored. Under certain conditions (see “Turn-On Conditions” in “Chapter 5 - Front Panel Operation” of the Operating Guide), location 0 may hold the device state that is automatically recalled at power turn-on.

The following power supply parameters are stored by **SAV**:

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>Parameters</th>
<th>Example</th>
<th>Query Syntax</th>
<th>Related Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR[:LEV][:IMM]</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>CURR:PROT:STAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIG:DATA[:VAL]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTP[:STAT]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTP:PROT:DEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTP:REL[:STAT]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTP:REL:POL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOLT[:LEV][:IMM]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOLT:PROT[:LEV]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Caution**

The power supply uses nonvolatile memory for recording register states. Programs that repeatedly use **SAV** for recalling states cause frequent write cycles to the memory and can eventually exceed the maximum number of write cycles for the memory (see in the power supply Operating Guide).

---

**SRE**

**Meaning and Type**

Service Request Enable  Device Interface

**Description**

This command sets the condition of the Service Request Enable Register. This register determines which bits from the Status Byte Register (see **STB** for its bit configuration) are allowed to set the Master Status Summary (MSS) bit and the Request for Service (RQS) summary bit. A 1 in any Service Request Enable Register bit position enables the corresponding Status Byte Register bit and all such enabled bits then are logically ORed to cause Bit 6 of the Status Byte Register to be set. See “Chapter 4 - Status Reporting” for more details concerning this process.

When the controller conducts a serial poll in response to SRQ, the RQS bit is cleared, but the MSS bit is not. When **SRE** is cleared (by programming it with 0), the power supply cannot generate an SRQ to the controller.
Command Syntax  *SRE <NRf>
Parameters  0 to 255
Default Value  (See *PSC)
Example  *SRE 20
Query Syntax  *SRE?
Returned Parameters  <NR1>  (Register binary value)
Related Commands  *ESE  *ESR  *PSC

Caution  If *PSC is programmed to $0$, then the *SRE command causes a write cycle to nonvolatile memory. The nonvolatile memory has a finite number of write cycles (see Supplemental Characteristics in the power supply Operating Guide). Programs that repeatedly write to nonvolatile memory can eventually exceed the maximum number of write cycles and may cause the memory to fail.

*STB?

Meaning and Type  
*STB  *Device Status

Description  
This query reads the Status Byte register, which contains the status summary bits and the Output Queue MAV bit. Reading the Status Byte register does not clear it. The input summary bits are cleared when the appropriate event registers are read (see “Chapter 4 - Status Reporting” for more information). The MAV bit is cleared at power on or by *CLS.

A serial poll also returns the value of the Status Byte register, except that bit 6 returns Request for Service (RQS) instead of Master Status Summary (MSS). A serial poll clears RQS, but not MSS. When MSS is set, it indicates that the power supply has one or more reasons for requesting service.

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>OPER</td>
<td>MSS$^1$</td>
<td>ESB</td>
<td>MAV</td>
<td>QUES</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(RQS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit Weight</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

ESB = Event status byte summary; MAV = Message available
MSS = Master status summary; OPER = Operation status summary;
QUES = Questionable status summary; RQS = Request for service

$^1$Also represents RQS.  $^2$These bits are always zero.

Query Syntax  *STB?
Returned Parameters  <NR1>  (Register binary value)
Related Commands  (None)
*TRG

Meaning and Type
Trigger Device Trigger

Description
This command generates a trigger when the trigger subsystem has BUS selected as its source. The command has the same affect as the Group Execute Trigger (<GET>) command.

Command Syntax *TRG
Parameters (None)
Query Syntax (None)
Related Commands ABOR CURR:TRIG INIT TRIG:IMM VOLT:TRIG

<TST>

Meaning and Type
Test Device Test

Description
This query causes the power supply to do a self-test and report any errors (see “Selftest Error Messages” in “Chapter 3 - Turn-On Checkout” of the power supply Operating Guide).

Query Syntax *TST?
Returned Parameters
0 Indicates power supply passed self-test.
Nonzero Indicates an error code.

Related Commands (None)

*WAI

Meaning and Type
Wait to Continue Device Status

Description
This command instructs the power supply not to process any further commands until all pending operations are completed. “Pending operations” are as defined under the *OPC command. *WAI can be aborted only by sending the power supply an HP-IB DCL (Device Clear) command.

Command Syntax *WAI
Parameters (None)
Query Syntax (None)
Related Commands *OPC *OPC?

DESCRIPTION OF SUBSYSTEM COMMANDS
Figure 3-2 is a tree diagram of the subsystem commands. Commands followed by a question mark (?) take only the query form. Except as noted in the syntax descriptions, all other commands take both the command and query form. The commands are listed in alphabetical order and the commands within each subsystem are grouped alphabetically under the subsystem.
**ABOR**

This command cancels any trigger actions presently in process. Pending trigger levels are reset equal to the their corresponding immediate values. **ABOR** also resets the WTG bit in the Operation Condition Status register (see “Chapter 4 - Status Reporting”). If INIT:CONT ON has been programmed, the trigger subsystem initiates itself immediately after ABORt, thereby setting WTG. **ABOR** is executed at power turn on and upon execution of *RCL or RST.

**Command Syntax**  
ABORt

**Parameters**  
(None)

**Examples**  
ABOR

**Query Syntax**  
(None)

**Related Commands**  
INIT  *RST  *TRG  TRIG

**Calibration Commands**

See Appendix A in the power supply Operating Guide

---

**Figure 3-2. Subsystem Commands Tree Diagram**

- **ROOT**
  - **ABOR**
    - **CALibrate** (Note 1)
    - **DISPlay** — [WINDow] — [STATE]
      - **INITiate** — [IMMediate] — [MODE]
        - **CONTinuous** — [TEXT] — [DATA]
        - **PROtection** — [STATe]
      - **DIgital** — [DATA] — [VALUE]
        - **PROtection** — [LEVEL]
    - **MEASure** — [CURRent] — [VOLTage] — [DC]?
      - **OUTPUT** — [STATe]
        - **PROtection** — [CLEar] — [DELay]
          - **RELay** — [STATe] — [POLarity]
        - **OPERation** — [EVENT]?
          - **CONDITION?**
          - **ENABLE**
          - **NTRansition**
          - **PTRANSition**
        - **PRESet**
          - **QUESTIONable**
            - **EVENT?**
              - **CONDITION?**
              - **ENABLE**
              - **NTRansition**
              - **PTRANSition**
        - **SYStem**
          - **ERror?**
            - **LANGuage**
            - **VERSion?**
        - **TRigger** — [IMMediate] — [SOURCE]
Current Subsystem
This subsystem programs the output current of the power supply.

CURR

CURR:TRIG
These commands set the immediate current level or the pending triggered current level of the power supply. The immediate level is the current programmed for the output terminals. The pending triggered level is a stored current value that is transferred to the output terminals when a trigger occurs. A pending triggered level is unaffected by subsequent CURR commands and remains in effect until the trigger subsystem receives a trigger or an ABORt command is given. If there is no pending CURR:TRIG level, then the query form returns the CURR level. In order for CURR:TRIG to be executed, the trigger subsystem must be initiated (see INITiate).

Command Syntax
[SOURCE]:CURRent[:LEVEL] [:IMMediate][:AMPLitude] <NRf+>
[SOURCE]:CURRent[:LEVEL]:TRIGgered [:AMPLitude] <NRf+>

Parameters
Table 3-1

Default Suffix
A

*RST Value
Table 3-1

Examples
CURR 200 MA  CURRENT:LEVEL 200 MA
CURRENT:LEVEL:IMMEDIATE:AMPLITUDE 2.5
CURR:TRIG 20  CURRENT:LEVEL:TRIGGERED 20

Query Syntax
[SOURCE]:CURRent[:LEVEL] [:IMMediate][:AMPLitude]?
[SOURCE]:CURRent[:LEVEL] [:IMMediate][:AMPLitude]? MAX
[SOURCE]:CURRent[:LEVEL] [:TRIGgered][:AMPLitude]?
[SOURCE]:CURRent[:LEVEL]:TRIGgered [:AMPLitude]? MAX
[SOURCE]:CURRent[:LEVEL]:TRIGgered [:AMPLitude]? MIN

Returned Parameters
<NR3>  CURR? and CURR:TRIG? return presently programmed immediate and triggered levels. If not triggered level is programmed, both returned values are the same.
CURR? MAX and CURR? MIN return the maximum and minimum programmable immediate current levels.
CURR:TRIG? MAX and CURR:TRIG? MIN return the maximum and minimum programmable triggered current levels.

Related Commands
For CURR *SAV *RCL *RST
For CURR:TRIG ABOR CURR *RST

CURR:PROT:STAT
This command enables or disables the power supply overcurrent protection (OCP) function. If the overcurrent protection function is enabled and the power supply goes into constant-current operation, then the output is disabled and the Questionable Condition status register OC bit is set (see “Chapter 4 - Status Reporting”). An overcurrent condition can be cleared with the OUTP:PROT:CLE command after the cause of the condition is removed.

Command Syntax
[SOURCE]:CURRent:PROTection:STATe <bool>

Parameters
0 | 1 | OFF | ON

*RST Value
OFF

Examples
CURR:PROT:STAT 0  CURRENT:PROTECTION:STATE OFF
CURR:PROT:STAT 1  CURRENT:PROTECTION:STATE ON

Query Syntax
[SOURCE]:CURRent:PROTection:STATe?

Returned Parameters
<NR1>  0 or 1

Related Commands
OUTP:PROT:CLE *RST
DIG:DATA
This command sets and reads the power supply digital control port when that port is configured for Digital I/O operation. Configuring of the port is done via an internal jumper (see Appendix D in the Operating Guide). The port has three signal pins and a digital ground pin. Pins 1 and 2 are output pins controlled by bits 0 and 1. Pin 3 is controlled by bit 3 and can be programmed to serve either as an input or as an output. Pin 4 is the digital ground.

Bit position 2 normally serves as an output. To change it to an input, it must first be programmed high. The DIG:DATA? query returns the last programmed value in bits 0 and 1 and the value read at pin 3 in bit 2. The bits are turned on and off in straight binary code as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Bit Configuration</th>
<th>Pin Configuration 1</th>
<th>Value</th>
<th>Bit Configuration</th>
<th>Pin Configuration 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0</td>
<td>Lo Lo Output Gnd</td>
<td>4</td>
<td>0 0 1</td>
<td>Lo Lo Input Gnd</td>
</tr>
<tr>
<td>1</td>
<td>1 0 0</td>
<td>Hi Lo Output Gnd</td>
<td>5</td>
<td>1 0 1</td>
<td>Hi Lo Input Gnd</td>
</tr>
<tr>
<td>2</td>
<td>0 1 0</td>
<td>Lo Hi Output Gnd</td>
<td>6</td>
<td>0 1 1</td>
<td>Lo Hi Input Gnd</td>
</tr>
<tr>
<td>3</td>
<td>1 1 0</td>
<td>Hi Hi Output Gnd</td>
<td>7</td>
<td>1 1 1</td>
<td>Hi Hi Input Gnd</td>
</tr>
</tbody>
</table>

1Pins 1 and 2 are always outputs

Command Syntax: [SOURce]:DIGital:DATA[:VALue] <NRf>
Parameters: 0 to 7
Suffix: (None)
*RST Value: 0
Examples: DIG:DATA 7 DIGITAL:DATA:VALUE 7
Query Syntax: [SOURce]:DIGital:DATA?
Returned Parameters: <NR1> Values from 0 to 7
Related Commands: *RST *RCL *SAV

Display Subsystem
This subsystem controls the state and output of the alphanumeric portion of the display.

DISP
Enables or disables the display. When disabled, the display characters are blank. The annunciators are not affected by this command.

Command Syntax: DISPlay[:WINDow][:STATe] <bool>
Parameters: 0 | 1 | OFF | ON
Suffix: (None)
*RST Value: ON
Examples: DISP ON DISPLAY:STATE ON
Query Syntax: DISPlay[:WINDow][:STATe]?
Returned Parameters: <NR1> 0 or 1
Related Commands: DISP:MODE DISP:TEXT *RST

DISP:MODE
Switches the display between its normal metering mode and a mode in which it displays text sent by the user. The command uses the character data <CRD> format.
| Command Syntax | DISPLAY[:WINDOW]:MODE NORM | TEXT |
| Parameters    | <CRD> NORMAL | TEXT |
| *RST Value    | NORM         |
| Examples      | DISPLAY:MODE NORM DISPLAY:MODE NORM |
| Query Syntax  | DISPLAY[:WINDOW]:MODE? |
| Returned Parameters | <CRD> NORMAL or TEXT |
| Related Commands | DISP DISP:TEXT *RST |

**DISP:TEXT**

Allows character strings to be sent to display. The characters will be displayed when the display mode is TEXT. The LCD has the following character set:

<table>
<thead>
<tr>
<th>character set</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uppercase letters</td>
<td>A through Z (Case-sensitive entry)</td>
</tr>
<tr>
<td>digits</td>
<td>0 through 9</td>
</tr>
<tr>
<td>punctuation</td>
<td>-</td>
</tr>
<tr>
<td>blank space</td>
<td></td>
</tr>
</tbody>
</table>

A display is capable of showing up to 12 characters. However, the three punctuation characters ([.,(,)] do not count toward the 12-character limit when they are preceded by an alphanumeric character. When punctuation characters are included, then the maximum number of characters (alphanumeric + punctuation) that can be displayed is 15. If it exceeds the display capacity, a message will be truncated to fit and no error message will be generated. If any character in the message is not a member of the above character set, the character will not be rejected but will be displayed as a “starburst” (all 16 segments of the character will light).

**Note**

*IEEE Standard Digital Interface for Programmable Instrumentation* requires that a string be enclosed in either single (’) or double (") quotes.

### Initiate Subsystem

This subsystem enables the trigger system. When a trigger is enabled, an event on a selected trigger source causes the specified triggering action to occur. If the trigger subsystem is not enabled, all trigger commands are ignored. If INIT:CONT is OFF, then INIT enables the trigger subsystem only for a single trigger action. The subsystem must be enabled prior to each subsequent trigger action. If INIT:CONT is ON, then the trigger subsystem is continuously enabled and INIT is redundant.
**INIT**

**Command Syntax**

```
INITiate[:IMMediate]
```

**Parameters**

For INIT[:IMM] (None)

For INIT:CONT 0|1|OFF|ON

**RST Value**

OFF

**Examples**

INIT INITiate:IMMEDIATE

INIT:CONT 1 INITiate:CONTinuous 1

**Query Syntax**

For INIT[:IMM] (None)

For INIT:CONT INIT:CONT?

**Returned Parameters**

<NRI> 0|1

**Related Commands**

ABOR <GET> *RST TRIG *TRG

---

**Measure Subsystem**

This query subsystem returns the voltage and current measured at the power supply's sense terminals.

**Query Syntax**

```
MEASURE:CURRent[:DC]? MEASURE:VOLTage[:DC]?
```

**Parameters**

(None)

**Default: Suffix**

A for MEAS:CURR? V for MEAS:VOLT?

**Examples**


**Returned Parameters**

<NR3>

---

**Output Subsystem**

This subsystem controls the power supply's voltage and current outputs and an optional output relay.

---

**Caution**

Do not install or program the HP Relay Accessories if the power supply maximum output current rating (see Table 3-1) exceeds the contact ratings of the relay.

---

**OUTP**

This command enables or disables the power supply output. The state of a disabled output is a condition of zero output voltage and a model-dependent minimum source current (see Table 3-1). The query form returns the output state.

**Command Syntax**

```
OUTPut[:STATe] <bool>
```

**Parameters**

0 | OFF | 1 | ON

**Suffix**

(None)

**RST Value**

0

**Examples**

OUTP 1 OUTPUT:STATE ON

**Query Syntax**

```
OUTPut[:STATe]?
```

**Returned Parameters**

<NRI> 0 or 1

**Related Commands**

*RST *RCL *SAV

---

**OUTP:PROT**

There are two output protection commands that do the following:

**OUTP:PROT:CLE**

Clears any OV (overvoltage), OC (overcurrent, unless set via external voltage control), OT (overtemperature), or RI (remote inhibit) protection features. After this command, the output is restored to the state it was in before the protection feature occurred.
**OUTP:PROT:DEL** Sets the time between the programming of an output change that produces a CV, CC, or UNREG condition and the recording of that condition by the Status Operation Condition register. The delay prevents the momentary changes in power supply status that can occur during reprogramming from being registered as events by the status subsystem. Since the delay applies to CC status, it also delays the OCP (overcurrent protection) feature. The OVP (overvoltage protection) feature is not affected by this delay.

**Examples**

```
OUTP:PROT:CLE  OUTPUT:PROTECTION:CLEAR
OUTPUT:PROTECTION:DELAY 75E-1
OUTP:PROT:DEL MIN  OUTPUT:PROT:DELAY MAX
```

**Query Syntax**

```
OUTP:PROT:CLE (None)
OUTP:PROT:DEL? MAX
```

**Returned Parameters**

```
<NR3> OUTP:PROT:DEL? returns value of programmed delay.
```

**Related Commands**

```
OUTP:PROT:CLE (None)
OUTP:PROT:DEL ^RST ^RCL ^SAV
```

---

**OUTP:REL**

**Caution**

Do not install or program the HP Relay Accessories if the power supply maximum output current rating (see Table 3-1) exceeds the contact ratings of the relay.

This command is valid only if the power supply is configured for the optional relay connector. Programming ON closes the relay contacts; programming OFF opens them. The relay is controlled independently of the output state. If the power supply is supplying power to a load, that power will appear at the relay contacts during switching. If the power supply is not configured for the relay option, sending either relay command generates an error.

**Command Syntax**

```
OUTP:REL [STATE] <bool>
```

**Parameters**

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

**^RST Value**

```
0
```

**Examples**

```
OUTP:REL 1  OUTP:REL OFF
```

**Query Syntax**

```
OUTP:REL?
```

**Returned Parameters**

```
0  | 1
```

**Related Commands**

```
OUTP:[STATE] ^RCL ^SAV
```

---

**OUTP:REL:POL**

**Caution**

Do not install or program the HP Relay Accessories if the power supply maximum output current rating (see Table 3-1) exceeds the contact ratings of the relay.

This command is valid only if the power supply is configured for the optional relay connector. Programming NORMal causes the relay output polarity to be the same as the power supply output. Programming REVerse causes the relay output polarity to be opposite to that of the power supply output. If OUTP:[STATE] = ON when either relay command is sent, the power supply output voltage is set to 0 during the time that the relays are changing polarity.
If the power supply is not configured for the relay option, sending either relay command generates an error.

**Command Syntax**
OUTPut:RELay:POLarity <CRD>

**Parameters**
NORMAL | REVerse

**RST Value**
NORM

**Examples**
OUTP : REL : POL NORM

**Query Syntax**
OUTP:put:RELay:POLarity?

**Returned Parameters**
NORM | REV

**Related Commands**
OUTP [:STAT] *RCL *SAV

### Status Subsystem

This subsystem programs the power supply status registers. The power supply has three groups of status registers; **Operation**, **Questionable**, and **Standard Event**. The Standard Event group is programmed with Common commands as described in “Chapter 4 - Status Reporting”. The Operation and Questionable status groups each consist of the Condition, Enable, and Event registers and the NTR and PTR filters.

### Status Operation Registers

The bit configuration of all Status Operation registers is shown in the following table:

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>15-12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Name</td>
<td>NU</td>
<td>NU</td>
<td>CC</td>
<td>NU</td>
<td>CV</td>
<td>NU</td>
<td>NU</td>
<td>WTG</td>
<td>NU</td>
<td>NU</td>
<td>NU</td>
<td>NU</td>
<td>CAL</td>
</tr>
<tr>
<td>Bit Weight</td>
<td>2048</td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

CAL = Interface is computing new calibration constants; CC = The power supply is in constant-current mode.

CV = The power supply is in constant voltage mode; NU = (Not used); WTG = The interface is waiting for a trigger.

**Note**
See “Chapter 4 - Status Reporting” for more explanation of these registers.

### STAT:OPER?

This query returns the value of the Operation Event register. The Event register is a read-only register which holds (latches) all events that are passed by the Operation NTR and/or PTR filter. Reading the Operation Event register clears it.

**Query Syntax**
STATUS:OPERation[+EVENT+]

**Parameters**
None

**Returned Parameters**
<NR1> (Register Value)

**Examples**
STAT:OPER? STATUS:OPERATIONAL:EVENT?

**Related Commands**
*CLS STAT:OPER:NTR STAT:OPER:PTR
STAT:OPER:COND?
This query returns the value of the Operation Condition register. That is a read-only register which holds the real-time (unlatched) operational status of the power supply.

Query Syntax  STAT:OPER:CONDITION?
Parameters  (None)
Examples  STAT:OPER:COND?  STAT:OPER:CONDITION?
Returned Parameters  <NR1>  (Register value)
Related Commands  (None)

STAT:OPER:ENAB
This command and its query set and read the value of the Operational Enable register. This register is a mask for enabling specific bits from the Operation Event register to set the operation summary bit (OPER) of the Status Byte register. This bit (bit 7) is the logical OR of all the Operational Event register bits that are enabled by the Status Operation Enable register.

Command Syntax  STAT:OPER:ENABLE <NRf>
Parameters  0 to 32727
Suffix  (None)
Default Value  0
Examples  STAT:OPER:ENAB 1312  STAT:OPER:ENAB 1
STAT:OPER:CONDITION?
Query Syntax  STAT:OPER:ENABLE?
Returned Parameters  <NR1>  (Register value)
Related Commands  STAT:OPER:EVEN

STAT:OPER NTR/Ptr Commands
These commands set or read the value of the Operation NTR (Negative-Transition) and PTR (Positive-Transition) registers. These registers serve as polarity filters between the Operation Enable and Operation Event registers to cause the following actions:

- When a bit in the Operation NTR register is set to 1, then a 1-to-0 transition of the corresponding bit in the Operation Condition register causes that bit in the Operation Event register to be set.
- When a bit of the Operation PTR register is set to 1, then a 0-to-1 transition of the corresponding bit in the Operation Condition register causes that bit in the Operation Event register to be set.
- If the same bits in both NTR and PTR registers are set to 1, then any transition of that bit at the Operation Condition register sets the corresponding bit in the Operation Event register.
- If the same bits in both NTR and PTR registers are set to 0, then no transition of that bit at the Operation Condition register can set the corresponding bit in the Operation Event register.

Note  Setting a bit in the value of the PTR or NTR filter can of itself generate positive or negative events in the corresponding Operation Event register.
STAT:OPER:NTR/PTR

Command Syntax
STATus:OPERtion:NTRnsition <NRf>
STATus:OPERtion:PTRnsition <NRf>

Parameters
0 to 32727

Default Value
(0)

Examples
STAT:OPER:NTR 32
STAT:OPER:PTR 1312

Query Syntax
STAT:OPER:NTR? STAT:OPER:PTR?

Returned Parameters
<NR1> (Register value)

Related Commands
STAT:OPER:ENAB

STAT:PRES

This command sets all defined bits in the Status Subsystem PTR registers and clears all bits in the subsystem NTR and Enable registers. STAT:OPER:PTR is set to 1313 and STAT:QUES:PTR is set to 1555.

Command Syntax
STATus:PRESet

Parameters
(0)

Examples
STAT: PRES STATUS: PRESET

Query Syntax
(0)

Related Commands
(0)

Status Questionable Registers

The bit configuration of all Status Questionable registers is as follows:

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>15-11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Name</td>
<td>NU</td>
<td>UNR</td>
<td>RI</td>
<td>NU</td>
<td>NU</td>
<td>NU</td>
<td>OT</td>
<td>NU</td>
<td>NU</td>
<td>OC</td>
<td>OV</td>
<td></td>
</tr>
<tr>
<td>Bit Weight</td>
<td>1024</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

NU = (Not used); OC = Overcurrent protection circuit has tripped.
OT = Overtemperature status condition exists; OV = Overvoltage protection circuit has tripped.
RI = Remote inhibit is active; UNR = Power supply output is unregulated.

Note: See “Chapter 4 - Status Reporting” for more explanation of these registers.

STAT:QUES?

This query returns the value of the Questionable Event register. The Event register is a read-only register which holds (latches) all events that are passed by the Questionable NTR and/or PTR filter. Reading the Questionable Event register clears it.

Query Syntax
STATus:QUESTionale:Event?

Parameters
(0)

Returned Parameters
<NR1> (Register Value)

Examples
STAT:QUES? STATUS:QUESTIONABLE:Event?

Related Commands

Subsystem Commands
STAT:QUES:COND

This query returns the value of the Questionable Condition register. That is a read-only register which holds the real-time (unlatched) questionable status of the power supply.

<table>
<thead>
<tr>
<th>Query Syntax</th>
<th>STAT:QUES:tionable:COND?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>(None)</td>
</tr>
<tr>
<td>Examples</td>
<td>STAT:QUES:COND? STAT:QUES:CONDITION?</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;NR1&gt; (Register value)</td>
</tr>
<tr>
<td>Related Commands</td>
<td>(None)</td>
</tr>
</tbody>
</table>

STAT:QUES:ENAB

This command and its query set and read the value of the Questionable Enable register. This register is a mask for enabling specific bits from the Questionable Event register to set the questionable summary bit (QUES) of the Status Byte register. This bit (bit 3) is the logical OR of all the Questionable Event register bits that are enabled by the Questionable Status Enable register.

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>STAT:QUES:tionable:ENAB &lt;NR1&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>0 to 32727</td>
</tr>
<tr>
<td>Suffix</td>
<td>(None)</td>
</tr>
<tr>
<td>Default Value</td>
<td>0</td>
</tr>
<tr>
<td>Examples</td>
<td>STAT:QUES:ENAB 20 STAT:QUES:ENAB 16</td>
</tr>
<tr>
<td>Query Syntax</td>
<td>STAT:QUES:tionable:ENAB?</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;NR1&gt; (Register value)</td>
</tr>
<tr>
<td>Related Commands</td>
<td>STAT:QUES?</td>
</tr>
</tbody>
</table>

STAT:QUES NTR/ PTR Commands

These commands allow you to set or read the value of the Questionable NTR (Negative-Transition) and PTR (Positive-Transition) registers. These registers serve as polarity filters between the Questionable Enable and Questionable Event registers to cause the following actions:

- When a bit of the Questionable NTR register is set to 1, then a 1-to-0 transition of the corresponding bit of the Questionable Condition register causes that bit in the Questionable Event register to be set.
- When a bit of the Questionable PTR register is set to 1, then a 0-to-1 transition of the corresponding bit in the Questionable Condition register causes that bit in the Questionable Event register to be set.
- If the same bits in both NTR and PTR registers are set to 1, then any transition of that bit at the Questionable Condition register sets the corresponding bit in the Questionable Event register.
- If the same bits in both NTR and PTR registers are set to 0, then no transition of that bit at the Questionable Condition register can set the corresponding bit in the Questionable Event register.

Note

Setting a bit in the PTR or NTR filter can of itself generate positive or negative events in the corresponding Questionable Event register.

---

3-20 Language Dictionary

Subsystem Commands
<table>
<thead>
<tr>
<th>STAT:QUES:NTR/Ptr</th>
<th>SYST:ERR?</th>
<th>SYS:LANG</th>
<th>SYST:VERS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Syntax</td>
<td>STAT:QUESTIONable:NTRsition &lt;NRf&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STAT:QUESTIONable:PTrsition &lt;NRf&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>0 to 32727</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suffix</td>
<td>(None)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default Value</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples</td>
<td>STAT:QUES:NTR 16</td>
<td>STATUS:QUESTIONABLE:PTR 512</td>
<td></td>
</tr>
<tr>
<td>Query Syntax</td>
<td>STAT:QUES:NTR?</td>
<td>STAT:QUES:PTR?</td>
<td></td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;NR1&gt; (Register value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Related Commands</td>
<td>STAT:QUES:ENAB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SYST:ERR?**
This query returns the next error number followed by its corresponding error message string from the remote programming error queue. The queue is a FIFO (first-in, first-out) buffer that stores errors as they occur. As it is read, each error is removed from the queue. When all errors have been read, the query returns 0, NO ERROR. If more errors are accumulated than the queue can hold, the last error in the queue will be -350, TOO MANY ERRORS (see Table 5-1 in “Chapter 5 - Error Messages” for other error codes).

You can use the power supply front panel Error key to read errors from the queue. Errors generated at the front panel are not put into the queue but appear immediately on the display.

<table>
<thead>
<tr>
<th>Query Syntax</th>
<th>SYST:ERR?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>(None)</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;NR1&gt;,&lt;SRD&gt;</td>
</tr>
<tr>
<td>Examples</td>
<td>SYST:ERR? SYSTEM:ERROR?</td>
</tr>
<tr>
<td>Related Commands</td>
<td>(None)</td>
</tr>
</tbody>
</table>

**SYST:LANG**
This command switches the interface between its SCPI (TMSL) command language and its compatibility language. The compatibility language is provided for emulation of older power supply systems and is described in Appendix B. Sending the command causes:

- The alternate language to become active and to be stored in nonvolatile memory.
- The power supply to reset to the state stored in Location 0.

If the power supply is shut off, it will resume operation in the last-selected language when power is restored.

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>SYST:LANG &lt;string&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>TMSL</td>
</tr>
<tr>
<td>Default Value</td>
<td>TMSL or last selected language</td>
</tr>
<tr>
<td>Examples</td>
<td>SYST:LANG TMSL SYSTEM:LANGUAGE COMPATIBILITY</td>
</tr>
<tr>
<td>Query Syntax</td>
<td>SYST:LANG?</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>&lt;CRD&gt; TMSL</td>
</tr>
<tr>
<td>Related Commands</td>
<td>(None)</td>
</tr>
</tbody>
</table>

**SYST:VERS?**
This query returns the SCPI version number to which the power supply complies. The returned value is of the form YYYY.V, where YYYY represents the year and V is the revision number for that year.

Subsystem Commands
Trigger Subsystem
This subsystem controls remote triggering of the power supply.

TRIG
When the trigger subsystem is enabled, TRIG generates a trigger signal. The trigger will then:

1. Initiate a pending level change as specified by CURR[:LEV]:TRIG or VOLT[:LEV]:TRIG.
2. Clear the WTG bit in the Status Operation Condition register.
3. If INIT:CONT has been given, the trigger subsystem is immediately re-enabled for subsequent triggers. As soon as it is cleared, the WTG bit is again set to 1.

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>TRIGger[:IMMEDIATE]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>(None)</td>
</tr>
<tr>
<td>Examples</td>
<td>TRIG TRIGGER:IMMEDIATE</td>
</tr>
<tr>
<td>Query Syntax</td>
<td>(None)</td>
</tr>
<tr>
<td>Related Commands</td>
<td>ABOR CURR:TRIG INIT *TRG VOLT:TRIG</td>
</tr>
</tbody>
</table>

TRIG:SOUR
This command selects the trigger source. Since the power supply has no other trigger source than the HP-IE bus, this command need not be used. It is included in the command set to provide programming compatibility with other instruments (such as the HP Electronic Load family) that may have more than one trigger source.

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>TRIGger:SOURce &lt;CRD&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>BUS</td>
</tr>
<tr>
<td>*EST Value</td>
<td>BUS</td>
</tr>
<tr>
<td>Examples</td>
<td>TRIG:SOUR BUS TRIGGER:SOURce BUS</td>
</tr>
<tr>
<td>Query Syntax</td>
<td>TRIGger:SOURce?</td>
</tr>
<tr>
<td>Returned Parameters</td>
<td>BUS</td>
</tr>
<tr>
<td>Related Commands</td>
<td>*RST *TRG TRIG[:IMM]</td>
</tr>
</tbody>
</table>

Voltage Subsystem
This subsystem programs the output voltage of the power supply.

VOLT
VOLT:TRIG
These commands set the immediate voltage level or the pending triggered voltage level of the power supply. The immediate level is the voltage programmed for the output terminals. The pending triggered level is a stored voltage value that is transferred to the output terminals when a trigger occurs. A pending triggered level is unaffected by subsequent VOLT commands and remains in effect until the trigger subsystem receives a trigger or an ABORt command is given. If there is no pending VOLT:TRIG level, then the query form returns the VOLT level. In order for VOLT:TRIG to be executed, the trigger subsystem must be initiated (see INITiate).
**VOLT:TRIG**

Command Syntax

[SOURce]:VOLTage:[LEVel]:IMMediate:AMPLitude <NRfe+>

[SOURce]:VOLTage:[LEVel]:TRIGgered:AMPLitude <NRfe+>

Parameters

Table 3-1

Default Suffix

V

*RST Value

Table 3-1

Examples

VOLT 200 MA VOLTAGE:LEVEL 200 MA
VOLT:TRIG 20 VOLTAGE:LEVEL:TRIGGERED 20

Query Syntax

[SOURce]:VOLTage:[LEVel]:IMMediate:AMPLitude?
[SOURce]:VOLTage:[LEVel]:IMMediate:AMPLitude? MAX
[SOURce]:VOLTage:[LEVel]:IMMediate:AMPLitude? MIN
[SOURce]:VOLTage:[LEVel]:TRIGgered:AMPLitude?
[SOURce]:VOLTage:[LEVel]:TRIGgered:AMPLitude? MAX
[SOURce]:VOLTage:[LEVel]:TRIGgered:AMPLitude? MIN

Returned Parameters

<NR3> VOLT? and VOLT:TRIG? return presently programmed immediate and triggered levels. If not triggered level is programmed, both returned values are the same.

VOLT? MAX and VOLT? MIN return the maximum and minimum programmable immediate voltage levels.

VOLT:TRIG? MAX and VOLT:TRIG? MIN return the maximum and minimum programmable triggered voltage levels.

Related Commands

For VOLT *SAV *RCL *RST
For VOLT:TRIG ABOR VOLT *RST

**VOLT:PROT**

This command sets the overvoltage protection (OVP) level of the power supply. If the output voltage exceeds the OVP level, then the power supply output is disabled and the Questionable Condition status register OV bit is set (see “Chapter 4 - Status Reporting”). An overvoltage condition can be cleared with the OUTP:PROT:CLE command after the condition that caused the OVP trip is removed. The OVP always trips with zero delay and is unaffected by the OUTP:PROT:DEL command.

Command Syntax

[SOURce]:VOLTage:PROTection:LEVel <NRfe+>

*Alternate Syntax

[SOURce]:VOLTage:PROTection:AMPLitude <NRfe+>

Parameters

Table 3-1

Default Suffix

V

*RST Value

MAX

Examples

VOLT:PROT 21.5 VOLT:PROT:LEV MAX
VOLTAGE:PROTECTION:LEVEL 145E-1

Query Syntax

[SOURce]:VOLTage:PROTection:LEVel?
[SOURce]:VOLTage:PROTection:LEVel? MIN
[SOURce]:VOLTage:PROTection:LEVel? MAX

Returned Parameters

<NR3> VOLT:PROT? returns presently programmed OVP level.

VOLT:PROT? MAX and VOLT:PROT? MIN return the maximum and minimum programmable OVP levels.

Related Commands

OUTP:PROT:CLE *RST *SAV *RCL

* Available to accommodate earlier power supply programs.
COMMAND SUMMARY
This summary lists all power supply subsystem commands in alphabetical order, followed by all common commands in alphabetical order. See Table 3-1 for the command parameters accepted by each power supply model.

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subsystem Commands</strong></td>
<td></td>
</tr>
<tr>
<td>ABOR</td>
<td>(none)</td>
</tr>
<tr>
<td>CAL</td>
<td>(See Appendix A in the Operating Manual)</td>
</tr>
<tr>
<td>[SOUR]:CURR[:LEV][:IMM][:AMPL]</td>
<td>&lt;NRF+&gt;</td>
</tr>
<tr>
<td>[SOUR]:CURR[:LEV][:IMM][:AMPL]?</td>
<td>(none)</td>
</tr>
<tr>
<td>[SOUR]:CURR[:LEV]:TRIG[:AMPL]</td>
<td>&lt;NRF+&gt;</td>
</tr>
<tr>
<td>[SOUR]:CURR[:LEV]:TRIG[:AMPL]?</td>
<td>(none)</td>
</tr>
<tr>
<td>[SOUR]:CURR:PROT:STAT</td>
<td>0</td>
</tr>
<tr>
<td>[SOUR]:CURR:PROT:STAT?</td>
<td>(none)</td>
</tr>
<tr>
<td>[SOUR]:DIG:DATA[:VAL]</td>
<td>&lt;NRF&gt;</td>
</tr>
<tr>
<td>[SOUR]:DIG:DATA[:VAL]?</td>
<td>(none)</td>
</tr>
<tr>
<td>DISP:WIND:MODE</td>
<td>NORM</td>
</tr>
<tr>
<td>DISP:WIND:MODE?</td>
<td>(none)</td>
</tr>
<tr>
<td>DISP[:WIND][:STAT]</td>
<td>0</td>
</tr>
<tr>
<td>DISP[:WIND][:STAT]?</td>
<td>(none)</td>
</tr>
<tr>
<td>DISP[:WIND]:TEXT[:DATA]</td>
<td>&lt;STR&gt;</td>
</tr>
<tr>
<td>DISP[:WIND]:TEXT[:DATA]?</td>
<td>(none)</td>
</tr>
<tr>
<td>INIT[:MM]</td>
<td>(none)</td>
</tr>
<tr>
<td>INIT:CONT</td>
<td>0</td>
</tr>
<tr>
<td>INIT:CONT?</td>
<td>(none)</td>
</tr>
<tr>
<td>MEAS:CURR[:DC]?</td>
<td>(none)</td>
</tr>
<tr>
<td>MEAS:VOLT[:DC]?</td>
<td>(none)</td>
</tr>
<tr>
<td>OUTP[:STAT]</td>
<td>0</td>
</tr>
<tr>
<td>OUTP[:STAT]?</td>
<td>(none)</td>
</tr>
<tr>
<td>OUTP:PROT:CLE</td>
<td>(none)</td>
</tr>
<tr>
<td>OUTP:PROT:DEL</td>
<td>0 to 32.767</td>
</tr>
<tr>
<td>OUTP:PROT:DEL?</td>
<td>(none)</td>
</tr>
<tr>
<td>OUTP:REL[:STAT]</td>
<td>0</td>
</tr>
<tr>
<td>OUTP:REL[:STAT]?</td>
<td>(none)</td>
</tr>
<tr>
<td>OUTP:REL:POL</td>
<td>NORM</td>
</tr>
<tr>
<td>OUTP:REL:POL?</td>
<td>(none)</td>
</tr>
</tbody>
</table>
### Command Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT:OPER:COND?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:OPER:ENAB</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>STAT:OPER:ENAB?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:OPER:EVEN?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:OPER:NTR</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>STAT:OPER:NTR?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:OPER:PTR</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>STAT:OPER:PTR?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:PRES</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:QUES:COND?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:QUES:ENAB</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>STAT:QUES:ENAB?</td>
<td>(none)</td>
</tr>
<tr>
<td>STAT:QUES:EVEN?</td>
<td>(none)</td>
</tr>
<tr>
<td>SYST:ERR?</td>
<td>(none)</td>
</tr>
<tr>
<td>SYST:LANG</td>
<td>TMSL</td>
</tr>
<tr>
<td>SYST:LANG?</td>
<td>(none)</td>
</tr>
<tr>
<td>SYST:VERS?</td>
<td>(none)</td>
</tr>
<tr>
<td>TRIG:[IIM]</td>
<td>(none)</td>
</tr>
<tr>
<td>TRIG:SOUR</td>
<td>BUS</td>
</tr>
<tr>
<td>TRIG:SOUR?</td>
<td>(none)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SOUR]:VOLT[:LEV]:[IIM]:[AMPL]</td>
<td>&lt;NRf+&gt;[suffix]</td>
</tr>
<tr>
<td>[SOUR]:VOLT[:LEV]:[IIM]:[AMPL]?</td>
<td>(none)</td>
</tr>
<tr>
<td>[SOUR]:VOLT[:LEV]:TRIG[:AMPL]</td>
<td>&lt;NRf+&gt;[suffix]</td>
</tr>
<tr>
<td>[SOUR]:VOLT[:LEV]:TRIG[:AMPL]?</td>
<td>(none)</td>
</tr>
<tr>
<td>[SOUR]:VOLT:PROT[:LEV]</td>
<td>&lt;NRf+&gt;[suffix]</td>
</tr>
<tr>
<td>[SOUR]:VOLT:PROT[:LEV]?</td>
<td>&lt;NRf+&gt;[suffix]</td>
</tr>
</tbody>
</table>

### Common Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CLS</td>
<td>(None)</td>
</tr>
<tr>
<td>*ESE</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>*ESE?</td>
<td>(None)</td>
</tr>
<tr>
<td>*ESR?</td>
<td>(None)</td>
</tr>
<tr>
<td>*IDN?</td>
<td>(None)</td>
</tr>
<tr>
<td>*OPC</td>
<td>(None)</td>
</tr>
<tr>
<td>*OPC?</td>
<td>(None)</td>
</tr>
<tr>
<td>*PSC</td>
<td>&lt;bool&gt;</td>
</tr>
<tr>
<td>*PSC?</td>
<td>(None)</td>
</tr>
<tr>
<td>*RCL</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>*RST</td>
<td>(None)</td>
</tr>
<tr>
<td>*SAV</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>*SRE</td>
<td>&lt;NRf&gt;</td>
</tr>
<tr>
<td>*SRE?</td>
<td>(None)</td>
</tr>
<tr>
<td>*STB?</td>
<td>(None)</td>
</tr>
<tr>
<td>*TRG</td>
<td>(None)</td>
</tr>
<tr>
<td>*TST?</td>
<td>(None)</td>
</tr>
<tr>
<td>*WAI</td>
<td>(None)</td>
</tr>
</tbody>
</table>

### PROGRAMMING PARAMETERS

Table 3-1 lists the programming parameters for each of the models.
Table 3-1. Power Supply Programming Parameters (see note)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>HP Model and Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR[:LEV] MAX and CURR[:LEV]:TRIG MAX (Programming range is 0 to MAX)</td>
<td>6641A 6642A 6643A 6644A 6645A</td>
</tr>
<tr>
<td>6651A 6652A 6653A 6654A 6655A</td>
<td></td>
</tr>
<tr>
<td>51.188 A 25.594 A 15.356 A 9.214 A 4.095 A</td>
<td></td>
</tr>
<tr>
<td>6671A 6672A 6673A 6674A 6675A</td>
<td></td>
</tr>
<tr>
<td>225.23 A 102.37 A 61.43 A 35.83 A 18.43 A</td>
<td></td>
</tr>
<tr>
<td>6680A 6681A 6682A 6683A 6684A</td>
<td></td>
</tr>
<tr>
<td>875 A 580 A 240 A 160 A 128 A</td>
<td></td>
</tr>
</tbody>
</table>

*RST Current Value

| 6641A 6642A 6643A 6644A 6645A | |
| 0.08 A 0.04 A 0.024 A 0.014 A 0.006 A | |
| 6651A 6652A 6653A 6654A 6655A | |
| 0.205 A 0.100 A 0.060 A 0.036 A 0.016 A | |
| 6671A 6672A 6673A 6674A 6675A | |
| 0.88 A 0.40 A 0.24 A 0.14 A 0.07 A | |
| 6680A 6681A 6682A 6683A 6684A | |
| 73.71 A 48.75 A 20.26 A 13.51 A 10.79 A | |

OUTP:PROT:DEL 0 to 32.727 s (MAX) for all models

*RST Value 200 ms for all models

VOLT[:LEV] MAX and VOLT[:LEV]:TRIG MAX (Programming range is 0 to MAX)

| 6641A 6642A 6643A 6644A 6645A | |
| 6651A 6652A 6653A 6654A 6655A | |
| 8.190 V 20.475 V 35.831 V 51.425 V 122.85 V | |
| 6680A 6681A 6682A 6683A 6684A | |
| 5.145 V 8.190 V 21.50 V 32.75 V 41.0 V | |

*RST Voltage Value 0 V for all models

VOLT:PROT MAX (Programming range is 0 to MAX)

| 6641A 6642A 6643A 6644A 6645A | |
| 6651A 6652A 6653A 6654A 6655A | |
| 8.8 V 22.0 V 38.5 V 66.0 V 132.0 V | |
| 6671A 6672A 6673A 6674A 6675A | |
| 10.0 V 24.0 V 42.0 V 72.0 V 144.0 V | |
| 6680A 6681A 6682A 6683A 6684A | |
| 6.25 V 10.0 V 25.2 V 38.4 V 48.0 V | |

*RST OVP value MAX for all models

Note For programming accuracy and resolution, see the Specifications and Supplemental Characteristics in the Operating Guide.