Programming Manual

MPS Series
Modular System DC Power Supplies

BK PRECISION

Contents

1 About this Manual 6
1.1 Intended Audience 6
1.2 Related Documents 6

2 Syntax Convention 7
2.1 Introduction 8
2.1.1 Types of SCPI Messages 8
2.2 Types of SCPI Commands 9
2.2.1 Common Commands 9
2.3 Syntax of Program Messages 10

3 SCPI Data Types 11
3.1 Character Data Types 12
3.2 <NR1> Value Data Type 12
3.3 <NR2> Value Data Type 12
3.4 <NRf> Value Data Type 13
3.5 Boolean Data Type 13

4 About Commands & Queries 14
4.1 How They are Listed 14
4.2 How They are Described 14
4.3 When can they be used? 14
4.4 Command Notation 14

5 Common SCPI Commands 15
5.1 *CLS 16
5.2 *ESE 17
5.3 *ESR? 17
5.4 *IDN? 17
5.5 *OPC 18
5.6 *OPC? 18
5.7 *PSC 18
5.8 *PSC? 18
5.9 *RCL 19
5.10 *SAV 19
5.11 *RST 19
5.12 *SRE 19
5.13 *SRE? 19
5.14 *STB? 20
5.15 *TRG 20

5.16 *TST? . . . 20
5.17 *WAI . . . 20

6 Current System 21
6.1 [SOURce]:CURRent[:LEVel]:[IMMediate] <NRf> . . . 22
6.2 [SOURce]:CURRent:PROTection:CLEar . . . 23
6.3 [SOURce]:CURRent:PROTection[:LEVel] <NR2> . . . 24
6.4 [SOURce]:CURRent:PROTection:STATe <boolean> . . . 24
6.5 [SOURce]:CURRent:PROTection:TRIPped? . . . 25
6.6 [SOURce]:CURRent:SLOPE <NR2> . . . 25

7 Voltage Subsystem 26
7.1 [SOURce]:VOLTage[:LEVel]:[IMMediate] <NRf> . . . 27
7.2 [SOURce]:VOLTage:PROTection:CLEar . . . 28
7.3 [SOURce]:VOLTage:PROTection[:LEVel] <NR2> . . . 29
7.4 [SOURce]:VOLTage:PROTection:TRIPped? . . . 29
7.5 [SOURce]:VOLTage:SENSe <boolean> . . . 30
7.6 [SOURce]:VOLTage:SLOPE <NR2> . . . 30
7.7 [SOURce:]VOLTage:MAXimum <NR2> . . . 30
7.8 [SOURce:]VOLTage:MINimum <NR2> . . . 32

8 Digital Subsystem 33
8.1 [SOURce:]DIGital:INPut:DATA? . . . 34
8.2 [SOURce:]DIGital:OUTPut:DATA <NR1> . . . 34
8.3 [SOURce:]DIGital:PIN<NR1>:FUNCtion <Character> . . . 35
8.4 [SOURce:]DIGital:PIN<NR1>:POLarity <Character> . . . 36

9 Display Subsystem 37
9.1 :DISPlay[:WINDow][:STATe] <character> . . . 37

10 Initiate Subsystem 38
10.1 INITiate[:IMMediate] <NR1> . . . 38
10.2 INITiate:CONTinuous <NR1> . . . 38

11 Instrument Subsystem 39
11.1 :INSTrument[:SELect] . . . 39

12 List Subsystem 40
12.1 ABORt <NR1> . . . 41
12.2 [SOURce:]LIST:CLEar . . . 41
12.3 [SOURce:]LIST:COUNt <NR1> . . . 42
12.4 [SOURce:]LIST:DELete . . . 42
12.5 [SOURce:]LIST:DWELl <NR1> . . . 43
12.6 [SOURce:]LIST:CURRent[:LEVel] <NRf> . . . 44
12.7 [SOURce:]LIST:VOLTage[:LEVel] <NRf> . . . 45

12.8 [SOURce:]LIST:NEXT<NR1> 46
12.9 [SOURce:]LIST:NUMBer <NR1> 47
12.10 [SOURce:]LIST:SAVE 47
12.11 [SOURce:]LIST:STATe? 48
12.12 [SOURce:]LIST:STEP <NR1> 48
12.13 [SOURce:]LIST:STEP:NUMBer <NR1> 49
12.14 [SOURce:]LIST:TERMinate:LAST <NR1> 50
12.15 [SOURce:]LIST:TOUTput:BOSTep:DATA <boolean> 51
12.16 [SOURce:]LIST:TOUTput:EOSTep:DATA <boolean> 52
12.17 Create and Save List 53

13 LXI Subsystem 55
13.1 LXI:IDENtify[:STATe] <boolean> 55

14 Measure Subsystem 56
14.1 MEASure[:SCALar]:ALL[:DC]? 57
14.2 MEASure[:SCALar]:ALLCHannel[:DC]? 57
14.3 MEASure[:SCALar]:CURRent[:DC]? 57
14.4 MEASure[:SCALar]:POWer[:DC]? 58
14.5 MEASure[:SCALar]:VOLTage[:DC]? 58

15 Output Subsystem 59
15.1 OUTPut[:STATe] <boolean> 60
15.2 OUTPut:ALL[:STATe] <boolean> 60
15.3 OUTPut:INHibit:MODE <Character> 61
15.4 OUTPut:PAIR <Character> 62
15.5 OUTPut:PON[:STATe] <Character> 63
15.6 OUTPut:PROTection:CLEar 63
15.7 OUTPut:MODe <character> 64
15.8 OUTPut:TIMer <boolean> 64
15.9 OUTPut:TIMer:COUNt <NR1> <NR1> <NR1> 65
15.10 OUTPut:DELay:RISE <NR1> 65
15.11 OUTPut:DELay:Fall <NR1> 66

16 Power Subsystem 67
16.1 [SOURce:]POWer:LIMit <NRf> 68
16.2 [SOURce:]POWer:LIMit:ALL <NRf>, <NRf>, <NRf>, <NRf> 69
16.3 [SOURce:]POWer:MAX <NRf> 69

17 Status Subsystem 70
17.1 STATus:PRESet 71
17.2 STATus:QUEStionable[:EVENt]? 71
17.3 STATus:QUEStionable:ENABle <NR1> 72
17.4 STATus:QUEStionable:[:EVENt]? 72

17.5 STATus:QUEStionable:ENABle <NR1> . . . 73
17.6 STATus:QUEStionable:ISUMmary<NR1>:CONDition? <NR1> . . . 73
17.7 STATus:QUEStionable:ISUMmary<NR1>[:EVENt]? . . . 74
17.8 STATus:QUEStionable:ISUMmary<NR1>:ENABle <NR1> . . . 74
17.9 STATus:OPERation[:EVENt]? . . . 75
17.10 STATus:OPERation:ENABle <NR1> . . . 75

18 System Subsystem 81
18.1 SYSTem:BEEP[:IMMediate] <boolean> . . . 82
18.2 SYSTem:BEEP:STATe <boolean> . . . 82
18.3 SYSTem:CHANnel[:COUNt]? . . . 83
18.4 SYSTem:CHANnel:MODel? . . . 83
18.5 SYSTem:CHANnel:MODel:ALL? . . . 83
18.6 SYSTem:CHANnel:SERIal? . . . 83
18.7 SYSTem:CHANnel:SERIal:ALL? . . . 84
18.8 SYSTem:CHANnel:VERSion? . . . 84
18.9 SYSTem:CHANnel:VERSion:ALL? . . . 84
18.10 SYSTem:COMMunicate:RLState <Characater> . . . 85
18.11 SYSTem:COMMunicate:TCPip:CONTrol? . . . 85
18.12 SYSTem:DATE <date> . . . 86
18.13 SYSTem:ERRor? . . . 86
18.14 SYSTem:LOCal . . . 86
18.15 SYSTem:PERSona:MANufacturer? . . . 87
18.16 SYSTem:PERSona:MODel? . . . 87
18.17 SYSTem:REMote . . . 87
18.18 SYSTem:RWLock . . . 87
18.19 SYSTem:SECurity:IMMediate . . . 88
18.20 SYSTem:VERSion? . . . 88
18.21 SYSTem:TIME <time> . . . 88

19 Trigger Subsystem 89
19.1 TRIGger:SEQuence:SOURce <NR1> . . . 89

About this Manual

This manual describes how to use the Standard Commands for Programming Instruments (SCPI) to communicate with the MPS Series.

1.1 Intended Audience

This document is designed for instrument programmers tasked with creating SCPI-based programs for the MPS Series.

1.2 Related Documents

Refer to the following documents for more information:

- MPS Series User's Manual. This manual describes the operation of the 9810 Series.
- Standard Commands for Programming Instruments (SCPI), Volume1-4, Version 1990.0 May 1999, SCPI Consortium, 2515 Camino del Rio South, Suite 340, San Diego, Ca 92108.
- IEEE Std 488.2-1992, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017-2394, USA (ISBN 1-55937-238-9)

Syntax Convention

2.1 Introduction 8
2.1.1 Types of SCPI Messages 8
2.2 Types of SCPI Commands 9
2.2.1 Common Commands 9
2.3 Syntax of Program Messages 10

2.1 Introduction

SCPI (Standard Commands for Programmable Instruments) serves as a universal programming language designed for electronic test and measurement instruments. It is grounded in the IEEE 488.1 and IEEE 488.2 standards. The MPS series aligns with the SCPI language and incorporates the IEEE 488.2 STD status structure.

The commands can be issued over VISA or socket using TCP port 5025.

2.1.1 Types of SCPI Messages

In order to program an MPS instrument, it is necessary to create a program message. This message comprises one or more appropriately formatted SCPI commands transmitted from the controller to the MPS instrument. The program message, which can be sent at any time, requests the instrument to execute a specific action or provide data or status information. These requests are also referred to as queries.

Upon receiving a query, the MPS instrument responds by sending a response message back to the controller. This response message contains data formatted in a specific SCPI format.

The following documents provide more information about SCPI programming:

- Standard Commands for Programming Instruments (SCPI), Volume1-4, Version 1990.0 May 1999, SCPI Consortium, 2515 Camino del Rio South, Suite 340, San Diego, Ca 92108.
- IEEE Std 488.2-1992, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017-2394, USA (ISBN 1-55937-238-9)

2.2 Types of SCPI Commands

Two types of SCPI commands are available: common commands, described below, and device-specific subsystem commands.

2.2.1 Common Commands

Common SCPI commands, as defined by IEEE 488.2, are responsible for controlling and managing generic system functions like reset, self-test, configuration storage, and device identification. Typically, common commands start with an asterisk (*), have a length of four to five characters, and may involve one or more parameters. The command keyword is separated from the initial parameter by a space. Multiple commands can be separated using a semicolon (;), as demonstrated below:

*RST; *CLS; *ESE 32; *OPC?

Refer to **Table 2.1** for a summary of these common SCPI commands applicable to programming the MPS series. For a detailed description of these commands, consult **Chapter 5**.

Command	Description
*CLS	Clears all Event Registers summarized in the status byte.
*ESE?	Returns an <NR1>, representing the value of the Standard Event Status Enable Register. Reading the value of the register will result in its clearance.
*ESR?	Returns an <NR1>, representing the value of the Standard Event Status Register. Reading the value of the register will result in its clearance.
*IDN?	Returns the unique identification string of the instrument.
*OPC?	Returns a "+1", when all pending selected device operations have been finished.
*RST	The Reset command performs a device reset. The Reset command is the third level of reset in a three-level reset strategy.
*SRE?	The Service Request Enable query allows the programmer to determine the current contents of the Service Request Enable Register.
*STB?	Returns the current binary value of the Status Byte Register.

Table 2.1 Common SCPI Commands

2.3 Syntax of Program Messages

A program message consist of one or more properly formatted SCPI commands, a parameter (if necessary, and a terminator sent from the controller to the MPS instrument to request some action or to query the instrument for a response.

Figure 2.3 shows the syntax of a program message:

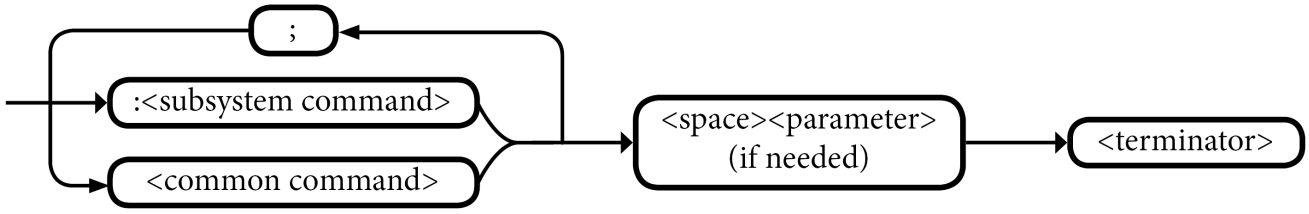


Figure 2.1 Syntax of Program Messages

A semicolon (;) is used to link commands from different groups.

SCPI Data Types

SCPI defines various data types for use in program messages and response messages.

The MPS series uses the following subset of SCPI data types:

- Character
- <NR1>
- <NR2>
- <NRf>
- <Boolean>

This section summarizes these data types. Refer to the SCPI standards document for more information about these data types.

3.1 Character Data Types

If a command parameter takes data type, a specific number of settings are allowed for the parameter.

Example In the command DISPlay:PAGE the user can specify one of the following character data types:

{ MEAS | MANUMEM | MANUCOMM | PROGMEM1 | PROGMEM2 | PROGMEM3 | PROGCOMM | SYSTENV | SYSTCOM | SYSTTOOL | INTF | EXTF}

Character data types have the following characteristics:

- Can be expressed in either the short or long form, while response messages return them exclusively in the short form.
- Are case insensitive in program messages but in response messages are standardized to uppercase.
- Must have a specific length.

3.2 <NR1> Value Data Type

The data type <NR1> is utilized to indicate zero, positive, and negative integer values, including optional signs.

The following values are examples of the <NR1> data types:

0 100 -10

3.3 <NR2> Value Data Type

The data type <NR2> is utilized to indicate zero, positive, and positive and negative decimal values, including optional signs and decimal points.

The difference between <NR1> and <NR2> is the explicit decimal point.

The following values are examples of the <NR1> data types:

200.50 100.0 0.0

NOTICE 0 is a special case and redundant decimal points are ignored.

3.4 <NRf> Value Data Type

The <NRf> data type is employed to define floating-point values. These values encompass digits with an implied decimal point, an explicit decimal point, or an explicit decimal point along with an exponent.

The following values are examples of the <NRf> data types:

200 15.000e-3 0.015

3.5 Boolean Data Type

A Boolean data type for a parameter and response represents a single binary condition that is either True or False. Boolean values are defined as follows:

- **0 or OFF** : Indicates that the condition is False.
- **1 or ON** : Indicates that the condition is True.

NOTICE The characters OFF and ON are not case sensitive.

About Commands & Queries

This section lists and describes the remote control commands and queries recognized by the instrument. All commands and queries can be executed in either local or remote state.

The description, command syntax, query syntax, example and respond can be found in a section. The commands are given in both long and short form. All examples are shown in short form. Queries perform actions such as obtaining information, and are recognized by the question mark (?) following the header.

4.1 How They are Listed

The commands are listed by subsystem and alphabetical order according to their short form.

4.2 How They are Described

In the descriptions themselves, a brief explanation of the function performed is given. This is followed by a presentation of the formal syntax, with the header given in Upper-and-Lower-Case characters and the short form derived from it in ALL UPPER-CASE characters. Where applicable, the syntax of the query is given with the format of its response.

4.3 When can they be used?

The commands and queries listed here can be used for the HVL seires.

4.4 Command Notation

The following notation is used in the commands:

< > Angular brackets enclose words that are used as placeholders, of which there are two types: the header path and the data parameter of a command.

:= A colon followed by an equals sign separates a placeholder from the description of the type and range of values that may be used in a command instead of the placeholder.

{ } Braces enclose a list of choices, one of which one must be made.

[ ] Square brackets enclose optional items.

… An ellipsis indicates that the items both to its left and right may be repeated a number of times.

Common SCPI Commands

IEEE standard defines the common commands used for querying the basic inSyntaxion of the instrument or executing basic operations. These commands usually start with "*" and the length of the keywords of the command is usually 3 characters.

5.1 *CLS 16
5.2 *ESE 17
5.3 *ESR? 17
5.4 *IDN? 17
5.5 *OPC 18
5.6 *OPC? 18
5.7 *PSC 18
5.8 *PSC? 18
5.9 *RCL 19
5.10 *SAV 19
5.11 *RST 19
5.12 *SRE 19
5.13 *SRE? 19
5.14 *STB? 20
5.15 *TRG 20
5.16 *TST? 20
5.17 *WAI 20

5.1 *CLS

Description This command clears all status data structures in a device. For a device which minimally complies with SCPI, these registers are:

SESR OPERation Status	**(IEEE 488.2)**
Register QUEStionable	**(SCPI)**
Status Register Error/Event	**(SCPI)**
Queue	**(SCPI)**

Execution of *CLS shall also clear any additional status data structures implemented in the device. The corresponding enable registers are unaffected.

***CLS** forces the device into OCIS and OQIS without setting the **No Operation Pending** flag TRUE and without setting the OPC bit of the SESR TRUE and without placing a “1” into the Output Queue.

For example, suppose a device implements **INITiate[:IMMediate]** as an overlapped command. Assuming that the trigger model is programmed so that it will eventually return to the IDLE state, and that INITiate[:IMMediate] takes longer to execute than *OPC, sending these commands to this device:

INITiate;*OPC

results in initiating the trigger model and, after some time, setting the OPC bit in the SESR. However, sending these commands:

INITiate;*OPC;*CLS

still initiates the trigger model. Since the operation is still pending when the device executes ***CLS**, the device does not set the OPC bit until it executes another ***OPC** command.

Example *CLS

5.2 *ESE

Description The Standard Event Status Enable command sets the Standard Event Status Enable Register bits

Syntax **Command** *ESE?

5.3 *ESR?

Description Query the Standard Event Status Register. Once a bit is set, it remains set until cleared by a ***CLS** (clear status) command or queried by this command. A query of this register returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

Syntax **Query** *ESR?

Example **Query** *ESR? Returns the standard event register

Related *CLS, *ESE

5.4 *IDN?

Description The *IDN? query causes the instrument to identify itself. The response comprises manufacturer, model, serial number, software version and firmware version.

Syntax **Query** *IDN?

Response *IDN, <device id>,<model>,<serial number>, <software version>, <hardware version>.

<device id>:=“BK” is used to identify instrument.

<model>:= A model identifier less than 14 characters will contain the model number.

<serial number>:= Number that uniquely identifies the instrument.

<firmware version>:= Firmware revision number.

<hardware version>:= Hardware revision number.

Example *IDN?

Returns: B&K Precision,MPS1102,XXXXXXXXX,0.90-1.00

5.5 *OPC

Description The Operation Complete command causes the device to generate the operation complete message in the Standard Event Status Register when all pending selected device operations have been finished.

Syntax **Command** *OPC

5.6 *OPC?

Description The Operation Complete query places an ASCII character “1” into the device’s Output Queue when all pending selected device operations have been finished.

Syntax **Query** *OPC?

Example **Query** *OPC? Query the operation complete register.

5.7 *PSC

Description The Power-On Status Clear command controls the automatic power-on clearing of the Service Request Enable Register, the Standard Event Status Enable Register, the Parallel Poll Enable Register, and device-specific event enable registers. This command may also affect the clearing of other status registers.

Syntax **Command** *PSC

5.8 *PSC?

Description The Power-On Status Clear query allows the programmer to query the device’s power-on-status-clear flag. A returned value of zero indicates that the Standard Event Status Enable Register, Service Request Enable Register, and the Parallel Poll Enable Register will retain their status when power is restored to the MPS. A returned value of one indicates that the registers listed above will be cleared when power is restored to the device.

Syntax **Query** *PSC?

Example **Query** *PSC?

5.9 *RCL

Description The *RCL command restores the current settings of a device from a copy stored in local memory.

Syntax **Command** *RCL <NR1>

Parameters <**NR1**> := {0 to 9}

Example **Command** *RCL 1 **Query** *RCL?

5.10 *SAV

Description The *SAV command stores the current settings of the device in local memory.

Syntax **Command** *SAV <NR1>

Parameters <**NR1**> := {0 to 9}

Example **Command** *SAV 2 **Query** *SAV?

5.11 *RST

Description The Reset command performs a device reset. The Reset command is the third level of reset in a three-level reset strategy.

Syntax **Query** *RST?

Example **Query** *RST?

5.12 *SRE

Description The Service Request Enable command sets the Service Request Enable Register bits.

Syntax **Command** *SRE

Example **Query** *SRE

5.13 *SRE?

Description The Service Request Enable query allows the programmer to determine the current contents of the Service Request Enable Register.

Syntax **Query** *SRE?

Example **Query** *SRE?

5.14 *STB?

Description The Read Status Byte query allows the programmer to read the status byte and Master Summary Status bit.

Syntax **Query** *STB?

Example **Query** *STB?

5.15 *TRG

Description The *TRG command generates an immediate trigger when the trigger source is set to BUS

Syntax **Command** *TRG

Example **Command** *TRG

5.16 *TST?

Description The self-test query causes an internal self-test and places a response into the Output Queue indicating whether or not the device completed the self-test without any detected errors.

Syntax **Query** *TST?

Example **Query** *TST?

5.17 *WAI

Description The Wait-to-Continue command prevents the device from executing any further commands or queries until the nooperation-pending flag is TRUE.

Syntax **Command** *WAI

Example **Command** *WAI

Current System

The [SOURce]:CURRent subsystem controls the current's amplitude characteristics of the source.

6.1 [SOURce]:CURRent[:LEVel]:[IMMediate] <NRf> 22
6.2 [SOURce]:CURRent:PROTection:CLEar 23
6.3 [SOURce]:CURRent:PROTection[:LEVel] <NR2> 24
6.4 [SOURce]:CURRent:PROTection:STATe <boolean> 24
6.5 [SOURce]:CURRent:PROTection:TRIPped? 25
6.6 [SOURce]:CURRent:SLOPE <NR2> 25

6.1 [SOURce]:CURRent[:LEVel]:[IMMediate] <NRf>

Description The command sets the immediate current level of the selected channel.

The query returns the programmed current level in the **<NR2>** data type.

Syntax **Command** [SOURce]:CURRent[:LEVel][:IMMediate] <NRf>

Query [SOURce]:CURRent[:LEVel][:IMMediate]?

Parameters <**NRf**> := { 0.000 to max current range }

MODEL	MPS1101 MPS1301	MPS1102 MPS1302	MPS1103 MPS1303	MPS1104 MPS1304
Current				
Range	0 - 20 A	0 - 9.5 A	0 - 5 A	0 - 3 A
Resolution	1 mA			

Table 6.1 Current Ranges

Example **Command** CURR 10.00 Sets the current level to 10 A.

Query CURR? Query the set current level.

6.2 [SOURce]:CURRent:PROTection:CLEar

Description The command causes the protection circuit to be cleared.

NOTICE This command is an event and has no association to the *RST condition.

Syntax **Command** [SOURce]:CURRent:PROTection:CLEar

Example **Command** CURR:PROT:CLE

6.3 [SOURce]:CURRent:PROTection[:LEVel] <NR2>

Description This command sets the current level at which the output protection circuit will trip.

The query returns the programmed protection current level in the **<NR2>** data type.

Syntax **Command** [SOURce]:CURRent:PROTection[:LEVel] <NR2>

Query [SOURce]:CURRent:PROTection[:LEVel]?

Parameters <**NR2**> := { 0.001 to max current range }

Example **Command** CURR:PROT 20
Query CURR:PROT?

6.4 [SOURce]:CURRent:PROTection:STATe <boolean>

Description The command enables/disabled the output's over current protection function.

The query returns the over current protection state, with the retuned value being "0" (OFF) or "1" (ON).

Syntax **Command** [SOURce]:CURRent:PROTection:STATe <boolean>

Query [SOURce]:CURRent:PROTection:STATe?

Parameters <**boolean**> := { 0 | 1 | OFF | ON }

Example **Command** CURR:PROT:STAT 1
Query CURR:PROT:STAT?

6.5 [SOURce]:CURRent:PROTection:TRIPped?

Description This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

The query returns if the protection cirucit is tripped, with the retuned value being "0" (untripped) or "1" (tripped).

Syntax

Command [SOURce]:CURRent:PROTection:TRIPped?

Query [SOURce]:CURRent:PROTection:TRIPped?

Example

Query CURR:PROT:TRIPped?

6.6 [SOURce]:CURRent:SLOPE <NR2>

Description Sets the slew rate of the output change when a new output level is programmed. The units are in A/sec.

The query returns the current's programmed slew rate. The returned value is in the **<NR2>** data type.

Syntax

Command [SOURce]:CURRent:SLOPe <NR2>

Query [SOURce]:CURRent:SLOPe?

Parameters <**NR2**> := { 0.1 to 5400 | MIN | MAX}

Example

Command CURR:SLOPe 100

Query CURR:SLOPe?

Voltage Subsystem

The [SOURce:]VOLTage subsystem controls the voltage's amplitude characteristics of the source.

7.1 [SOURce]:VOLTage[:LEVel]:[IMMediate] <NRf> 27
7.2 [SOURce]:VOLTage:PROTection:CLEar 28
7.3 [SOURce]:VOLTage:PROTection[:LEVel] <NR2> 29
7.4 [SOURce]:VOLTage:PROTection:TRIPped? 29
7.5 [SOURce]:VOLTage:SENSe <boolean> 30
7.6 [SOURce]:VOLTage:SLOPE <NR2> 30
7.7 [SOURce:]VOLTage:MAXimum <NR2> 30
7.8 [SOURce:]VOLTage:MINimum <NR2> 32

7.1 [SOURce]:VOLTage[:LEVel]:[IMMediate] <NRf>

Description The command sets the immediate voltage level of the selected channel.

The query returns the programmed voltage level in the **<NR2>** data type.

Syntax **Command** [SOURce]:VOLTage[:LEVel][:IMMediate] <NRf>

Query [SOURce]:VOLTage[:LEVel][:IMMediate]?

Parameters <**NRf**> := { 0.000 to max voltage range }

MODEL	MPS1101 MPS1301	MPS1102 MPS1302	MPS1103 MPS1303	MPS1104 MPS1304
Voltage				
Range	0 - 15 V	0 - 32 V	0 - 60 V	0 - 100 V
Resolution	1 mV			

Table 7.1 Voltage Range

Example **Command** VOLT 10.00

Query VOLT?

7.2 [SOURce]:VOLTage:PROTection:CLEar

Description The command causes the protection circuit to be cleared.

NOTICE This command is an event and has no association to the *RST condition.

Syntax **Command** [SOURce]:VOLTage:PROTection:CLEar

Example **Command** VOLT:PROT:CLE

7.3 [SOURce]:VOLTage:PROTection[:LEVel] <NR2>

Description This command sets the voltage level at which the output protection circuit will trip.

The query returns the programmed protection voltage level in the **<NR2>** data type.

Syntax

Command [SOURce]:VOLTage:PROTection[:LEVel] <NR2>

Query [SOURce]:VOLTage:PROTection[:LEVel]?

Parameters <**NR2**> := { 0.001 to max voltage range }

Example

Command VOLT:PROT 20

Query VOLT:PROT?

7.4 [SOURce]:VOLTage:PROTection:TRIPped?

Description This query returns a 1 if the protection circuit is tripped and a 0 if it is untripped.

The query returns if the protection cirucit is tripped, with the retuned value being "0" (untripped) or "1" (tripped).

Syntax

Command [SOURce]:VOLTage:PROTection:TRIPped?

Query [SOURce]:VOLTage:PROTection:TRIPped?

Example

Query VOLT:PROT:TRIPped?

7.5 [SOURce]:VOLTage:SENSe <boolean>

Description The command enables/disabled the output's sense function.

The query returns the sense state, with the retuned value being "0" (OFF) or "1" (ON).

Syntax **Command** [SOURce]:VOLTage:SENSe <boolean>

Query [SOURce]:VOLTage:SENSe?

Parameters <**boolean**> := { 0 | 1 | OFF | ON }

Example **Command** VOLT:SENS 1
Query VOLT:SENS?

7.6 [SOURce]:VOLTage:SLOPE <NR2>

Description Sets the slew rate of the output change when a new output level is programmed. The units are in A/sec.

The query returns the voltage's programmed slew rate. The returned value is in the <**NR2**> data type.

Syntax **Command** [SOURce]:VOLTage:SLOPe <NR2>

Query [SOURce]:VOLTage:SLOPe?

Parameters <**NR2**> := { 0.1 to 5400 | MIN | MAX}

Example **Command** VOLT:SLOPe 100
Query VOLT:SLOPe?

7.7 [SOURce:]VOLTage:MAXimum <NR2>

Description The command sets the maximum configurable voltage level. Units are in volts.

The query returns the programmed maximum configurable voltage level. The data is retuned in the <**NR2**> data type.

Syntax **Command** [SOURce:]VOLTage:MAXimum <NR2>

Query [SOURce:]VOLTage:MAXimum?

Parameters <**NR2**> := { 0.000 to max voltage range | MIN | MAX }

MODEL	MPS1101 MPS1301	MPS1102 MPS1302	MPS1103 MPS1303	MPS1104 MPS1304
Voltage				
Range	0 - 15 V	0 - 32 V	0 - 60 V	0 - 100 V
Resolution	1 mV			

Table 7.2 Voltage Range

Example **Command** VOLT:MAX 100.00
Query VOLT:MAX?

7.8 [SOURce:]VOLTage:MINimum <NR2>

Description The command sets the minimum configurable voltage level. Units are in volts.

The query returns the programmed minimum configurable voltage level. The data is retuned in the **<NR2>** data type.

Syntax **Command** [SOURce:]VOLTage:MINimum <NR2>

Query [SOURce:]VOLTage:MINimum?

Parameters <**NR2**> := { 0.000 to min voltage range | MIN | MAX }

MODEL	MPS1101 MPS1301	MPS1102 MPS1302	MPS1103 MPS1303	MPS1104 MPS1304
Voltage				
Range	0 - 15 V	0 - 32 V	0 - 60 V	0 - 100 V
Resolution	1 mV			

Table 7.3 Voltage Range

Example **Command** VOLT:MIN 10.00

Query VOLT:MIN?

Digital Subsystem

The Digital Subsystem allows for configuration of the digital functions. This subsystem allows users to control, query, and configure digital parameters such as digital output lines, trigger settings.

8.1 [SOURce:]DIGital:INPut:DATA? 34
8.2 [SOURce:]DIGital:OUTPut:DATA <NR1> 34
8.3 [SOURce:]DIGital:PIN<NR1>:FUNCtion <Character> 35
8.4 [SOURce:]DIGital:PIN<NR1>:POLarity <Character> 36

8.1 [SOURce:]DIGital:INPut:DATA?

Description The query returns the state of the digital control port. The data is returned in the binary-weighted value of the state of pins 1 through 5 in bits 0 through 4 respectively.

Syntax **Query** [SOURce:]DIGital:INPut:DATA?

Example **Query** DIG:INP:DATA?

8.2 [SOURce:]DIGital:OUTPut:DATA <NR1>

Description The command sets the state of the digital control port. This only affects the pins whose function has been set to Digital IO Operation. The port has 5 sinal pins and a digtal ground pin.

The query returns the set state of the digital control port. The value is returned in the **<NR1>** data type.

Syntax **Command** [SOURce:]DIGital:OUTPut:DATA <NR1>

Query [SOURce:]DIGital:OUTPut:DATA?

Parameters <**NR1**> := { 0 to 31 }
In the binary-weighted value that is written to the port, the pins are controlled according to the following bit assigments:

Pin	1	2	3	4	5
Bit Number	0	1	2	3	4
Decimal Number	1	2	4	8	16

Table 8.1 Binary-Weighted Value

NOTICE Bit values corresponding to digital port pins that are not configured as DIO are ignored.

Example **Command** DIG:OUTP:DATA 8
Query DIG:OUTP:DATA?

8.3 [SOURce:]DIGital:PIN<NR1>:FUNCtion <Character>

Description The command sets the function of the selected pin. The functions are saved in non-volatile memory.

The query returns the set function of the selected pin. The functioned is returned in the **<character>** data type.

Syntax

Command [SOURce:]DIGital:PIN<NR1>:FUNCtion <Character>

Query [SOURce:]DIGital:PIN<NR1>:FUNCtion?

Parameters <Character> := { NONE | DIO | DINPut | FAULt | INHibit | ONCouple | OFFCouple | TINPut | TOUTput }

Function	Definition
NONE	Disables pin.
DOUT	Digital output.
DINPut	Digital input-only mode.
FAULt	Provides an alert signal when the power supply detects an error or abnormal condition.
INHibit	Allows external devices to disable or "inhibit" the MPS's output.
TINPut	A trigger input function.
TOUTput	A trigger output function
SOUT	Sync output signal.
SINP	Sync input signal.

Table 8.2 Pin Functions

Example

Command DIG:PIN2:FUNC DIO

Query DIG:PIN2:FUNC?

8.4 [SOURce:]DIGital:PIN<NR1>:POLarity <Character>

Description The command sets the polarity of the selected pin.

The query returns the set polarity of the selected pin. The functioned is returned in the **<character>** data type.

Syntax

Command [SOURce:]DIGital:PIN<NR1>:POLarity <Character>

Query [SOURce:]DIGital:PIN<NR1>:POLarity?

Parameters <Character> := { POSitive | NEGative }

Polarity	Definition
POSitive	Logical true signal is a voltage high at pin. For trigger inputs and outputs, POSitive means rising edge.
NEGative	Logical true signal is a voltage low at pin. For trigger inputs and outputs, NEGative means falling edge.

Table 8.3 Pin Polarity

Example

Command DIG:PIN1:POL POS

Query DIG:PIN1:POL?

Display Subsystem

The DISPlay subsystem controls the selection and presentation of textual, graphical, and TRACe information. This information includes measurement data, user-interaction displays, and data presented to the instrument by the controller. DISPlay is independent of, and does not modify, how data is returned to the controller

9.1 :DISPlay[:WINDow][:STATe] <character> 37

9.1 :DISPlay[:WINDow][:STATe] <character>

Description The command enables/disables the selected window.

The query returns the state of the set display page. The returned value is in the <**boolean**> data type.

Syntax **Command** :DISPlay[:WINDow][:STATe] <**character**>

Query :DISPlay[:WINDow][:STATe]?

Parameters <**boolean**> := { 0 | 1 | OFF | ON }

Example **Command** DISP ON

Query DISP?

Initiate Subsystem

The INITiate subsystem allows users to issue commands that trigger specific actions, such as starting the list function.

10.1 INITiate[:IMMediate] <NR1> 38
10.2 INITiate:CONTinuous <NR1> 38

10.1 INITiate[:IMMediate] <NR1>

Description The command issues the trigger to run the selected programmed list. The channel is a selected through the <NR1> parameter. Refer to section OUTPut:MODE <NR1> for more information on selecting a list.

Syntax **Command** INITiate[:IMMediate]

Parameters <**NR1**> := { 0 to 3 }

Example **Command** INIT

10.2 INITiate:CONTinuous <NR1>

Description The command issues the trigger to run the selected programmed list with a continuous trigger enabled. The channel is a selected through the <NR1> parameter. Refer to section OUTPut:MODE <NR1> for more information on selecting a list. The command requires the user to specify the channel and state, where <NR1> will assign the channel and <boolean> the state.

Syntax **Command** INITiate:CONTinuous **<NR1>**,**<bool>**

Query INITiate:CONTinuous?

Parameters <**NR1**> := { 0 to 3 }

<**boolean**> := { 0 | 1 | ON | OFF }

Example **Command** INIT:CONT

Instrument Subsystem

The INSTrument subsystem provides a mechanism to identify and select channels through a number. The INSTrument Identifiers have no fixed correspondence to the numeric suffixes allowed with the command headers.

11.1 :INSTrument[:SELect] 39

11.1 :INSTrument[:SELect]

Description This command selects a channel. When a channel is selected all other channels are unavailable for programming until selected. By default channel one is selected.

Syntax **Command** :INSTrument[:SELect] <NR1>

Query :INSTrument[:SELect]?

Parameters **<NR1>** := {0 |1 |2}

0 := Channel 1
1 := Channel 2
2 := Channel 3
3 := Channel 4

NOTICE The programmable channels begin at 0. Therefore, to program channel 1 select 0, to program channel 2 select 1, to program channel 3 selected 2, and to program channel 4 select 3.

Query INSTrument[:SELect]?

Example INST 2

Response INST?

List Subsystem

The LIST subsystem controls automatic sequencing through associated lists of specified signal values.

12.1 ABORt <NR1> 41
12.2 [SOURce:]LIST:CLEar 41
12.3 [SOURce:]LIST:COUNt <NR1> 42
12.4 [SOURce:]LIST:DELete 42
12.5 [SOURce:]LIST:DWELl <NR1> 43
12.6 [SOURce:]LIST:CURRent[:LEVel] <NRf> 44
12.7 [SOURce:]LIST:VOLTage[:LEVel] <NRf> 45
12.8 [SOURce:]LIST:NEXT<NR1> 46
12.9 [SOURce:]LIST:NUMBer <NR1> 47
12.10 [SOURce:]LIST:SAVE 47
12.11 [SOURce:]LIST:STATe? 48
12.12 [SOURce:]LIST:STEP <NR1> 48
12.13 [SOURce:]LIST:STEP:NUMBer <NR1> 49
12.14 [SOURce:]LIST:TERMinate:LAST <NR1> 50
12.15 [SOURce:]LIST:TOUTput:BOSTep:DATA <boolean> 51
12.16 [SOURce:]LIST:TOUTput:EOSTep:DATA <boolean> 52
12.17 Create and Save List 53

12.1 ABORt <NR1>

Description The command resets the trigger system of the selected channel and places all trigger sequences in the IDLE state. The channel is a selected through the <NR1> parameter. The list will not resume until the INITiate[:IMMediate] command is sent.

Syntax **Command** ABORt

Parameters <**NR1**> := { 0 to 3 }

Example **Command** ABORt Abort the list function.

12.2 [SOURce:]LIST:CLEar

Description The command clears all the step and List Run parameters of the selected list. To clear a list the list configuration mode must first be enabled by sending the command [SOURce:]LIST:NUMBer <NR1>.

Syntax **Command** [SOURce:]LIST:CLEar

Parameters none

Example **Command** LIST:NUMB 2;:LIST:CLE;:LIST:SAVE Clears parameters of list 2.

12.3 [SOURce:]LIST:COUNt <NR1>

Description The command configures the count parameter of the selected list. The count parameter dictates the number of repetitions the list will undergo. The list can be programmed to run infinitely by using the parameter INFinity. The command can only be used once the **Edit List** mode is enabled. Refer to section [SOURce:]LIST:NUMBer <NR1>.

The query returns the programmed count for the selected list in the <NR1> data type.

NOTICE Changes made in **Edit List** will not be saved until the command [SOURce:]LIST:SAVELIST:SAVE] is sent.

Syntax **Command** [SOURce:]LIST:MODe <NR1>

Query [SOURce:]LIST:MODe?

Parameters <**NR1**> := { 0 to 100000 | INFinity }

Example

Command	LIST:COUN 0	Configures the list to run once.
Query	LIST:COUN?	Query the list's run count.

12.4 [SOURce:]LIST:DELete

Description The command deletes the selected step and all subsequent steps. For example if a list contains 10 steps and LIST:DELete 2, steps 2 through 10 are deleted.

Syntax **Command** [SOURce:]LIST:DELete **<NR1>**

Parameters <**NR1**> := { 0 to 512 }

Example

Command	LIST:NUMB 2;:LIST:DEL 2;:LIST:SAVE	Deletes step 2 and all subsequent steps.

12.5 [SOURce:]LIST:DWELI <NR1>

Description The command sets the dwell time of the step whose **Edit Step** mode is currently active. Units are in seconds. Configuring the step's dwell time requires the instrument to be in **Edit Step** mode initially. Refer to section **[SOURce:]LIST:STEP:NUMBer <NR1>**

The query returns the set dwell time of the step whose **Edit Step** mode is currently active. The returned value is in the <NRf> data type.

Syntax

Command	[SOURce:]LIST:DWELI <NRf>
Query	[SOURce:]LIST:STEP:DWELI?

Parameters <**NR1**> := { 0 to 9999 s }

Example

Command	LIST:STEP:DWEL 10	Sets the dwell time
Query	LIST:STEP:DWEL?	Query the step's set dwell time

12.6 [SOURce:]LIST:CURRent[:LEVel] <NRf>

Description The command sets the current level of the step whose **Edit Step** mode is currently active. Configuring the step's current level requires the instrument to be in **Edit Step** mode initially. Refer to section **[SOURce:]LIST:STEP:NUMBer <NR1>**

The query returns the set current level of the step whose **Edit Step** mode is currently active. The returned value is in the <NRf> data type.

Syntax **Command** [SOURce:]LIST:CURRent[:LEVel] <NRf>

Query [SOURce:]LIST:CURRent[:LEVel]?

Parameters <**NRf**> := { 0 to max current range }

MODEL	MPS1101 MPS1301	MPS1102 MPS1302	MPS1103 MPS1303	MPS1104 MPS1304
Current				
Range	0 - 20 A	0 - 9.5 A	0 - 5 A	0 - 3 A
Resolution	1 mA			

Table 12.1 Current Range

Example **Command** LIST:CURR 10

Query LIST:CURR?

12.7 [SOURce:]LIST:VOLTage[:LEVel] <NRf>

Description The command sets the voltage level of the step whose **Edit Step** mode is voltagely active. Configuring the step's voltage level requires the instrument to be in **Edit Step** mode initially. Refer to section **[SOURce:]LIST:STEP:NUMBer <NR1>**

The query returns the set voltage level of the step whose **Edit Step** mode is voltagely active. The returned value is in the <NRf> data type.

Syntax **Command** [SOURce:]LIST:VOLTage[:LEVel] <NRf>

Query [SOURce:]LIST:VOLTage[:LEVel]?

Parameters <**NRf**> := { 0 to max voltage range }

MODEL	MPS1101 MPS1301	MPS1102 MPS1302	MPS1103 MPS1303	MPS1104 MPS1304
Voltage				
Range	0 - 15 V	0 - 32 V	0 - 60 V	0 - 100 V
Resolution	1 mV			

Table 12.2 Voltage Range

Example **Command** LIST:VOLT 10

Query LIST:VOLT?

12.8 [SOURce:]LIST:NEXT<NR1>

Description The command configures the list to be executed once the selected list elapses. This allows for chaining of multiple list. The command can only be used once the **Edit List** mode is enabled. Refer to section [SOURce:]LIST:NUMBer <NR1>.

The query returns the programmed list to be executed upon completion of the current list. The value is returned in the <NR1> data type.

NOTICE Changes made in **Edit List** will not be saved until the command [SOURce:]LIST:SAVELIST:SAVE] is sent.

Syntax

Command	[SOURce:]LIST:NEXT <NR1>
Query	[SOURce:]LIST:NEXT?

Parameters <**NR1**> := { 1 to 10 }

Example

Command	LIST:NEXT 1	Configures list 1 as the next list.
Query	LIST:COUN?	Query the list that will be executed next.

12.9 [SOURce:]LIST:NUMBer <NR1>

Description The command enables **Edit List** mode for the selected list. When edit list mode is enabled the list's; **Mode**, **Next**, and **Repeat** parameters can be configured. The instrument can enter **Edit Step** mode once Edit List mode has been enabled.

The query returns the programmed list number in the <NR1> data type.

NOTICE Changes made in **Edit List** will not be saved until the command [SOURce:]LIST:SAVELIST:SAVE] is sent.

Syntax **Command** [SOURce:]LIST:NUMBer <NR1>

Query [SOURce:]LIST:NUMBer?

Parameters <**NR1**> := { 1 to 10 }

Example

Command	LIST:NUMB 1	Enables **Edit List** mode for list 1.
Query	LIST:NUMB?	Query the selected list.

12.10 [SOURce:]LIST:SAVE

Description The command saves all set parameters of the list being configured. Prior to sending the save command all changes are stored in temporary memory.

Syntax **Command** [SOURce:]LIST:SAVE

Parameters none

Example

Command	LIST:SAVE	Saves all list changes.

12.11 [SOURce:]LIST:STATe?

Description The query returns the running state of the loaded list for the active channel in the <boolean> data type.

Mode	Description
0	Idle
1	Active

Table 12.3 Active State

Syntax **Query** [SOURce:]LIST:STATe?

Example **Query** LIST:STATe? Returns the list's running state.

12.12 [SOURce:]LIST:STEP <NR1>

Description The command configures the triggering mode (pace) of the set list.
When the pace is set to **Dwell** the next step will be initiated once the dwell time of the current step has elapsed.
When the pace is set to **Trigger** the list will remain in the current step even after the dwell time has elapsed. To proceed to the next step the user must input a trigger signal.

NOTICE The required trigger signal will vary based on the list's programmed trigger source.

The query returns the programmed paced in the <NR1> data type.

Syntax **Command** [SOURce:]LIST:STEP <NR1>

Query [SOURce:]LIST:STEP?

Parameters <**NR1**> := { 0 to 1 }

Mode	Description
0	Dwell
1	Trigger

Table 12.4 List Trigger Mode

Example **Command** LIST:STEP 1 Sets the list's pace to trigger.
Query LIST:STEP? Query the selected pace.

12.13 [SOURce:]LIST:STEP:NUMBer <NR1>

Description The command enables **Edit Step** mode for the designated step, allowing for configuration of parameters such as **Voltage, Current, BOST, EOST, and Dwell**. Accessing **Edit Step** requires the instrument to be in **Edit List** mode initially. Refer to section **[SOURce:]LIST:NUMBer <NR1>**

The query returns the step currently being modified. The returned value is in the <NR1> data type.

Syntax

Command	[SOURce:]LIST:STEP:NUMBer <NR1>
Query	[SOURce:]LIST:STEP:NUMBer?

Parameters <**NR1**> := { 0 to 100 }

Example

Command	LIST:STEP:NUMB: 1	Enables **Edit Step** mode for step 1
Query	LIST:STEP:NUMB?	Query the configurable step.

12.14 [SOURce:]LIST:TERMinate:LAST <NR1>

Description The command configures the **After List** parameter which determines the load's input level upon the completion of any list.
When the after list state is set to **"Last"**, the input's level will retain the last list value.
When the after list state is set to **"DC"**, the output parameters will revert to the DC value that was in effect before the commencement of the list.

The query returns the configured after list parameter in the <NR1> data type.

Syntax

Command [SOURce:]LIST:TERMinate:LAST <NR1>

Query [SOURce:]LIST:TERMinate:LAST?

Parameters <**NR1**> := { 0 | 1 }

Mode	Description
0	DC
1	Last

Table 12.5 After List Mode

Example

Command	LIST:TERM:LAST 1	Sets the list's after list to Last.
Query	LIST:TERM:LAST?	Query the set after list parameter.

12.15 [SOURce:]LIST:TOUTput:BOSTep:DATA <boolean>

Description The command enables/disabled a TTL trigger-out signal at the beginning of the selected step (BOST). Configuring the BOST requires the instrument to be in **Edit Step** mode initially. Refer to section **[SOURce:]LIST:STEP:NUMBer <NR1>**

The query returns the state of the BOST for the step whose **Edit Step** mode is currently active. The returned value is in the <boolean> data type.

Syntax

Command [SOURce:]LIST:TOUTput:BOSTep:DATA <boolean>

Query [SOURce:]LIST:TOUTput:BOSTep:DATA?

Parameters <**boolean**> := { 0 | 1 }

Mode	Description
0	OFF
1	ON

Table 12.6 BOST State

Example

Command	LIST:TOUT:BOST:DATA 1	Enables BOST
Query	LIST:TOUT:BOST:DATA?	Query the BOST state

12.16 [SOURce:]LIST:TOUTput:EOSTep:DATA <boolean>

Description The command enables/disabled a TTL trigger-out signal at the end of the selected step (EOST). Configuring the EOST requires the instrument to be in **Edit Step** mode initially. Refer to section **[SOURce:]LIST:STEP:NUMBer <NR1>**

The query returns the state of the EOST for the step whose **Edit Step** mode is currently active. The returned value is in the <boolean> data type.

Syntax

Command [SOURce:]LIST:TOUTput:EOSTep:DATA <boolean>

Query [SOURce:]LIST:TOUTput:EOSTep:DATA?

Parameters <**boolean**> := { 0 | 1 }

Mode	Description
0	OFF
1	ON

Table 12.7 EOST State

Example

Command	LIST:TOUT:EOST:DATA 1	Enables EOST
Query	LIST:TOUT:EOST:DATA?	Query the EOST state

12.17 Create and Save List

Editing a list requires commands to be sent in a specific sequence. Some commands necessitate that the instrument be in a specific mode. If a command is sent when the instrument is not in the required mode, the instrument will beep, indicating it does not recognize the command.

The list sequence can be broken down into 5 seperate modes.

1. **Enable List Configuration**
2. **Configure List Parameters**
3. **Enable Step Configuration**
4. **Configure Step Paramters**
5. **Save List**

List Example

Enable List Configuration

1. **ABORT** Abort all running list to avoid any conflicts
2. **INSTrument[:SELect] <NR1>** Selects the module that will be running the created list
3. **[SOURce:]LIST:NUMBer <NR1>** Selects the list memory slot to be edited

Configure List Parameters

4. **[SOURce:]LIST:COUNt <NR1>** Sets the number of repetitions the list will undergo
5. **[SOURce:]LIST:NEXT <NR1>** Sets the list to run once the current list completes

Enable Step Configuration

6. **[SOURce:]LIST:STEP:NUMBer <NR1>** Selects the step in the list to be edited

Configure Step Parameters

7. **[SOURce:]LIST:VOLTage[:LEVel] <NRf>** Sets the voltage level of the selected step
8. **[SOURce:]LIST:CURRent[:LEVel] <NRf>** Sets the current level of the selected step
9. **[SOURce:]LIST:TOUTput:BOSTep:DATA <boolean>**
   Enables/Disables TTL trigger-out signal at the begging of the selected step
10. **[SOURce:]LIST:TOUTput:EOSTep:DATA <boolean>**
    Enables/Disables TTL trigger-out signal at the end of the selected step
11. **[SOURce:]LIST:DWELl <NR1>** Sets the dwell time of the selected step

Save List

12. **[SOURce:]LIST:SAVE** Saves changes made to the selected list

LXI Subsystem

The LXI subsystem controls the front panel LXI identify indicator on or off. When turned on, the "LAN" status indicator on the front panel blinks on and off to identify the instrument that is being addressed.

13.1 LXI:IDENtify[:STATe] <boolean> 55

13.1 LXI:IDENtify[:STATe] <boolean>

Description The command sets and the status of the LXI LAN status indicator on the LAN Status dialog.

The query returns the state of the LXI LAN status. The returned value is in the <**boolean**> data type.

Syntax **Command** :LXI:IDENtify[:STATe] <**boolean**>

Query :LXI:IDENtify[:STATe]?

Parameters <**boolean**> := { 0 | 1 | OFF | ON }

State		Description
0	OFF	Changes the LXI Status indicator to 'NORMAL' and closes the dialog if it was opened by this command.
1	ON	Changes the LXI Status indicator to 'IDENTIFY' and opens the dialog if it was not already open.

Table 13.1 LXI IdentifyState

Example **Command** LXI:IDEN ON

Query LXI:IDEN?

Measure Subsystem

The MEASure subsystem acquires data from the power supply using a set of high-level instructions.

14.1 MEASure[:SCALar]:ALL[:DC]? 57
14.2 MEASure[:SCALar]:ALLCHannel[:DC]? 57
14.3 MEASure[:SCALar]:CURRent[:DC]? 57
14.4 MEASure[:SCALar]:POWer[:DC]? 58
14.5 MEASure[:SCALar]:VOLTage[:DC]? 58

14.1 MEASure[:SCALar]:ALL[:DC]?

Description The query returns the averaged measured input voltage, current, and power of the selected channel. The averaging parameter allows for the adjustment of the sample count utilized for computing the average values.

Values are returned in the < data type.

Syntax **Query** MEASure[:SCALar]:ALL[:DC]?

Parameters none

Example **Query** MEAS:ALL?

14.2 MEASure[:SCALar]:ALLCHannel[:DC]?

Description The query returns the averaged measured input voltage, current, and power of all active channel. The averaging parameter allows for the adjustment of the sample count utilized for computing the average values.

Values are returned in the < data type.

Syntax **Query** MEASure[:SCALar]:ALLCHannel[:DC]?

Parameters none

Example **Query** MEAS:ALLCH?

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

Description The query returns the averaged measured input current. The averaging parameter allows for the adjustment of the sample count utilized for computing the average values.

Values are returned in the < data type, with units in amperes.

Syntax **Query** MEASure[:SCALar]:CURRent[:DC]?

Parameters none

Example **Query** MEAS:CURR?

14.4 MEASure[:SCALar]:POWer[:DC]?

Description The query returns the averaged measured input power.

Values are returned in the < data type, with units in watts.

Syntax **Query** MEASure[:SCALar]:POWer[:DC]?

Parameters none

Example **Query** MEAS:POW?

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

Description The query returns the averaged measured input voltage. The averaging parameter allows for the adjustment of the sample count utilized for computing the average values.

Values are returned in the < data type, with units in volts.

Syntax **Query** MEASure[:SCALar]:VOLTage[:DC]?

Parameters none

Example **Query** MEAS:VOLT?

Output Subsystem

The Output subsystem controls the output state, power-on, protection, and delay functions.

15.1 OUTPut[:STATe] <boolean> 60
15.2 OUTPut:ALL[:STATe] <boolean> 60
15.3 OUTPut:INHibit:MODE <Character> 61
15.4 OUTPut:PAIR <Character> 62
15.5 OUTPut:PON[:STATe] <Character> 63
15.6 OUTPut:PROTection:CLEar 63
15.7 OUTPut:MODe <character> 64
15.8 OUTPut:TIMer <boolean> 64
15.9 OUTPut:TIMer:COUNt <NR1> <NR1> <NR1> 65
15.10 OUTPut:DELay:RISE <NR1> 65
15.11 OUTPut:DELay:Fall <NR1> 66

15.1 OUTPut[:STATe] <boolean>

Description The command enables/disables the output of the selected channel. The state of a disabled output is a condition of zero output voltage and zero source current.

The query returns the output state of the selected channel, with the returned value being "0" (OFF) or "1" (ON).

NOTICE Upon resetting the instrument, the output state is off.

Syntax
Command OUTPut[:STATe] **<boolean>**
Query OUTPut[:STATe]?

Parameters <Boolean> := { 0 | 1 | OFF | ON }

Example
Command OUTP 1
Query OUTP?

15.2 OUTPut:ALL[:STATe] <boolean>

Description The command enables/disables the output of all channels. The state of a disabled output is a condition of zero output voltage and zero source current.

NOTICE Upon resetting the instrument, the output state of all channels is set to off.

Syntax
Command OUTPut:ALL[:STATe] **<boolean>**

Parameters <Boolean> := { 0 | 1 | OFF | ON }

Example
Command OUTP:ALL 1

15.3 OUTPut:INHibit:MODE <Character>

Description The command sets the operating mode of the remote inhibit digital pin. The inhibit function shuts down the output in response to an external signal on the Inhibit input pin. The Inhibit mode is stored in non-volatile memory.
See **[SOURce:]DIGital:PIN<NR1>:FUNCtion <Character>**.

The query returns the set inhibit mode, the response is in the <Character> data type.

Syntax **Command** OUTPut:INHibit:MODE <Character>

Query OUTPut:INHibit:MODE?

Parameters <Character> := { LATChing | LIVE | OFF }

Mode	Description
LATChing	A logic-true signal on the Inhibit input causes the output state to latch OFF. The output remains disabled until the Inhibit input is returned to logic-false and the latched INH status bit is cleared by sending the **OUTPut:PROTection:CLEar** command or a protection clear command from the front panel.
LIVE	Allows the enabled output to follow the state of the Inhibit input. When the Inhibit input is true, the output is disabled. When the Inhibit input is false, the output is re-enabled.
OFF	The inhibit input is ignored

Table 15.1 Inhibit Modes

Example **Command** OUTP:INH:MODE LIVE
Query OUTP:INH:MODE?

15.4 OUTPut:PAIR <Character>

Description The command sets the operation mode to either OFF(normal) | PARA(Parallel) | SERI (Series) | TRAC(Track). For the command to be valid, the mainframe must have the same module model numbers installed in the corresponding channels. For example, if the pair value is 2, CH1 and CH2 must have matching model number modules. If the pair value is 3, CH1, CH2, and CH3 must have matching model number modules. Likewise, if the pair value is 4, CH1, CH2, CH3, and CH4 must have matching model number modules.

The query returns the set pairing mode. The response is in the <Character> data type.

Syntax **Command** OUTPut:PAIR <Character>

Query OUTPut:PAIR?

Parameters <Character> := { OFF | PARA<NR1> | SERI<NR1> | TRAC<NR1> }

Mode	Description
OFF	Normal operation mode, no modules are paired.
PARA	Pairs the selected modules in parallel.
SERI	Pairs the selected modules in series.
TRAC	Pairs the selected modules for tracking mode.

Table 15.2 Pair Modes

<NR1> := { 2 | 3 | 4 }

Value	Description
2	Pairs modules 1 and 2.
3	Pairs modules 1, 2, and 3.
4	Pairs all modules.

Table 15.3 Pair Value

Example **Command** OUTP:PAIR 1

NOTICE The return string will be in the NR1 data type. with 0 = OFF; 1= SERI2; 2 = SERI3; 3= SERI4; 4 = PARA2; 5 = PARA3; 6 = PARA4; 7 = TRAC2; 8 = TRAC3; 9 = TRAC4

15.5 OUTPut:PON[:STATe] <Character>

Description The command configures the power-on state. When USER is selected the memory location must be specified. Instrument states can be stored using the *SAV command.

Syntax

Command OUTPut:PON[:STATe] <Character>

Query OUTPut:PON[:STATe]?

Parameters <**Character**> := { Off | LAST | USER,<**NR1**>}

<**NR1**> := memory lcoation := { 0 to 9}

State	Description
OFF	Disables power on function.
LAST	Restores the settings configured before the most recent power cycle.
USER,<**NR1**>	Loads the saved settings in set memory location.

Table 15.4 Power ON State

Example

Command OUTP:PON RST

Query OUTP:PON?

15.6 OUTPut:PROTection:CLEar

Description The command clears the latched protection status that disables the output when a protection condition occurs. The output is restored to the state it was in before the protection condition occurred

Syntax

Command OUTPut:PROTection:CLEar

Query OUTPut:PROTection:CLEar

Parameters none

Example

Command OUTP:PROT:CLE

Query OUTP:PROT:CLE

15.7 OUTPut:MODe <character>

Description Configures the operation mode of the instruments. If List mode is selected the list to be executed must be specified in the command's parameter.

The query returns the electronic load's operation mode in the <character> data type.

Syntax

Command OUTPut:MODe <character>

Query OUTPut:MODe?

Parameters <**character**> := { FIX | LIST,<NR1> }

Example

Command OUTP:MOD LIST,1

Query OUTP:MOD?

15.8 OUTPut:TIMer <boolean>

Description The command enables/disables the output timer.

The query returns the set state of the output timer.The value is returned in the **<boolean>** data type.

Syntax

Command OUTPut:TIMer **<boolean>**

Query OUTPut:TIMer?

Parameters <**boolean**> := { 0 | 1 | ON | OFF }

Example

Command OUTP:TIM ON

Query OUTP:TIM?

15.9 OUTPut:TIMer:COUNt <NR1> <NR1> <NR1>

Description The command sets the dwell time of the output timer.

The query returns the set time of the output timer.Three values are returned in the **<NR1>** data type. The first being the hours count, followed by the minutes count, and the seconds count.

Syntax

Command OUTPut:TIMer:COUNt **<NR1>**,**<NR1>**,**<NR1>**

Query OUTPut:TIMer:COUNt?

Parameters <**NR1**> := { 0 to 99 seconds}

<**NR1**> := { 0 to 59 seconds}

<**NR1**> := { 0 to 59 seconds}

Example

Command OUTP:TIM:COUN 1,5,10

Query OUTP:TIM:COUN?

15.10 OUTPut:DELay:RISE <NR1>

Description The command specifies the delay in seconds that the instrument waits before turning the output on (rise). This allows multiple channels to turn on in sequence. The output will not turn on until its delay time has elapsed. This command affects on-to-off state transitions. It does NOT affect transitions to off caused by protection functions. :

The query returns the set time for the output delay rise.The value is returned in the **<NR1>** data type.

Syntax

Command OUTPut:DELay:RISE **<NR1>**

Query OUTPut:DELay:RISE?

Parameters <**NR1**> := { 0 to 1023 seconds}

Example

Command OUTP:DEL:RISE 10

Query OUTP:DEL:RISE?

15.11 OUTPut:DELay:Fall <NR1>

Description The command specifies the delay in seconds that the instrument waits before turning the output off (fall). This allows multiple channels to turn off in sequence. The output will not turn off until its delay time has elapsed. This command affects on-to-off state transitions. It does NOT affect transitions to off caused by protection functions.

The query returns the set time for the output delay rise.The value is returned in the **<NR1>** data type.

Syntax

Command	OUTPut:DELay:Fall **<NR1>**
Query	OUTPut:DELay:Fall?

Parameters <**NR1**> := { 0 to 1023 seconds}

Example

Command	OUTP:DEL:Fall 10
Query	OUTP:DEL:Fall?

Power Subsystem

The power subsystem controls and configures the power-related functions of the MPS serires.

16.1 [SOURce:]POWer:LIMit <NRf> 68
16.2 [SOURce:]POWer:LIMit:ALL <NRf>, <NRf>, <NRf>, <NRf> 69
16.3 [SOURce:]POWer:MAX <NRf> 69

16.1 [SOURce:]POWer:LIMit <NRf>

Description The command configures the maximum power limit of the active channel. The maximum power will vary based on the module rated power and the maximum power dirstibution of the mainframe.

NOTICE

If the combination of the modules' rated power exceed the mainframe's rated power the modules will reallocate the power limit to meet the mainframe's maximum rated power.

The query returns the set power limit of the active channel.

Values are returned in the < data type.

Syntax

Command [SOURce:]POWer:LIMit **<NRf>**

Query [SOURce:]POWer:LIMit?

Parameters

MPS130X := <**NRf**> := { 0 to 306 }

MPS110X := <**NRf**> := { 0 to 102 }

:= <**character**> := { MINimum | MAXimum}

Example

Command POW:LIM 306

Query POW:LIM?

16.2 [SOURce:]POWer:LIMit:ALL <NRf>, <NRf>, <NRf>, <NRf>

Description The command configures the maximum power limit of all channels. The maximum power will vary based on the module rated power and the maximum power dirstibution of the mainframe.

NOTICE If the combination of the modules' rated power exceed the mainframe's rated power the modules will reallocate the power limit to meet the mainframe's maximum rated power.

The query returns the set power limit of all channels.

Values are returned in the < data type in the format CH1 limit, CH2 limit, CH3 limit, CH4 limit

Syntax

Command [SOURce:]POWer:LIMit:ALL **<NRf>**, **<NRf>**, **<NRf>**, **<NRf>**

Query [SOURce:]POWer:LIMit:ALL?

Parameters MPS130X := <**NRf**> := { 0 to 306 }

MPS110X := <**NRf**> := { 0 to 102 }

Example

Command POW:LIM:ALL 306, 306, 102, 102

Query POW:LIM:ALL?

16.3 [SOURce:]POWer:MAX <NRf>

Description The query returns the mainframe's maximum power available.
[halfline] MPS1001 @ nominal 100 - 120 VAC: 600 W
[halfline] MPS1001 @ nominal 200 - 240 VAC: 1200 W

The value is returned in the < data type.

Syntax

Query [SOURce:]POWer:MAX?

Parameters none

Example

Query POW:MAX?

Status Subsystem

The STATus subsystem controls the SCPI-defined stastus-reporting structures (QUEStionable, OPERation, Instrument SUMmary and registers). These registers are comprimised of condition registers, an event register, and and enable register.

The queue provides a human readable record of instrument events. The application programmer may individually enable events into the queue. STATus:PRESet enables errors and disables all other events. If the summary of the queue is reported, it shall be reported in bit 2 of the status byte register.

17.1 STATus:PRESet 71
17.2 STATus:QUEStionable[:EVENt]? 71
17.3 STATus:QUEStionable:ENABle <NR1> 72
17.4 STATus:QUEStionable:[:EVENt]? 72
17.5 STATus:QUEStionable:ENABle <NR1> 73
17.6 STATus:QUEStionable:ISUMmary<NR1>:CONDition? <NR1> 73
17.7 STATus:QUEStionable:ISUMmary<NR1>[:EVENt]? 74
17.8 STATus:QUEStionable:ISUMmary<NR1>:ENABle <NR1> 74
17.9 STATus:OPERation[:EVENt]? 75
17.10 STATus:OPERation:ENABle <NR1> 75

17.1 STATus:PRESet

Description The PRESet command affects only the enable register and queue enabling for the status data structures. PRESet does not affect the Status Byte or the Standard Event status. PRESet does not clear any of the event registers or any item from the error/event queue. The *CLS command is used to clear all event registers and queues in the device status-reporting system.

Syntax **Command** STATus:PRESet

Parameters none

Example **Command** STAT:PRES

17.2 STATus:QUEStionable[:EVENt]?

Description The query returns the event register for the Questionable Status group. This is a read-only register, which stores (latches) all events register events. Reading the Questionable Status Event register clears it. The returned value is in the **<NR1>** data format.

Syntax **Query** STATus:QUEStionable[:EVENt]?

Parameters none

Example **Query** STAT:QUES?

17.3 STATus:QUEStionable:ENABle <NR1>

Description The command sets the bits in the enable register for the Questionable Status group. The enable register is a mask for enabling specific bits from the Operation Event register to set the QUES (questionable summary) bit of the Status Byte register. STATus:PRESet clears all bits in the enable register.

The query returns the bits in the enable register for the Questionable Status group. The binary weighted value returned is in the **<NR1>** data type.

Syntax

Command	STATus:QUEStionable:ENABle **<NR1>**
Query	STATus:QUEStionable:ENABle?

Parameters **<NR1>** := { 1023 }

Example

Command	STAT:QUES:ENAB 24
Query	STAT:QUES:ENAB?

17.4 STATus:QUEStionable:[:EVENt]?

Description The query returns the event register for the Questionable Instrument Status group. This is a read-only register, which stores (latches) all events register events. Reading the Questionable Status Event register clears it. The returned value is in the **<NR1>** data format.

Syntax

Query	STATus:QUEStionable:[:EVENt]?

Parameters none

Example

Query	STAT:QUES:?

17.5 STATus:QUEStionable:ENABle <NR1>

Description The command sets the bits in the enable register for the Questionable Instrument Status group. The enable register is a mask for enabling specific bits from the Operation Event register to set the QUES (questionable summary) bit of the Status Byte register. STATus:PRESet clears all bits in the enable register.

The query returns the bits in the enable register for the Questionable Status group of the selected module. The binary weighted value returned is in the **<NR1>** data type.

Syntax

Command STATus:QUEStionable:ENABle **<NR1>**

Query STATus:QUEStionable:ENABle?

Parameters **<NR1>** := { 31 }

Example

Command STAT:QUES:ENAB 24

Query STAT:QUES:ENAB?

17.6 STATus:QUEStionable:ISUMmary<NR1>:CONDition? <NR1>

Description The query returns the CV or CC condition of the specified module. Reading the condition register is nondestructive. The response is NR1: 1 indicating CC mode and 2 indicating CV mode.

Syntax

Query STATus:QUEStionable:ISUMmary<NR1>:CONDition??

Parameters **<NR1>** := { 1 | 2 | 3 | 4 } (Module Selection)

Example

Query STAT:QUES:ISUM2:COND?

17.7 STATus:QUEStionable:ISUMmary<NR1>[:EVENt]?

Description The query returns the event register for the Questionable Instrument Isummary Status group of the selected module. This is a read-only register, which stores (latches) all events register events. Reading the Questionable Status Event register clears it. The returned value is in the **<NR1>** data format.

Syntax **Query** STATus:QUEStionable:ISUMmary<NR1>[:EVENt]?

Parameters **<NR1>** := { 1 | 2 | 3 | 4 } (Module Selection)

Example **Query** STAT:QUES:ISUMmary?

17.8 STATus:QUEStionable:ISUMmary<NR1>:ENABle <NR1>

Description The command sets the bits in the enable register for the Questionable Instrument Isummary Status group of the selected module. The enable register is a mask for enabling specific bits from the Operation Event register to set the QUES (questionable summary) bit of the Status Byte register. STATus:PRESet clears all bits in the enable register.

The query returns the bits in the enable register for the Questionable Status group of the selected module. The binary weighted value returned is in the **<NR1>** data type.

Syntax **Command** STATus:QUEStionable:<NR1>:ENABle **<NR1>**

Query STATus:QUEStionable:ISUMmary<NR1>:ENABle?

Parameters **<NR1>** := { 1 | 2 | 3 | 4 } (Module Selection)
<NR1> := { 1023 }

Example **Command** STAT:QUES:ISUMmary1:ENAB 24
Query STAT:QUES:ISUMmary1:ENAB?

17.9 STATus:OPERation[:EVENt]?

Description Queries the event register for the Operation Status group. This is a read-only register. Reading the Operation Status Event register clears it. The returned value is in the **<NR1>** data format.

Syntax **Query** STATus:OPERation[:EVENt]?

Parameters none

Example **Query** STAT:OPER?

17.10 STATus:OPERation:ENABle <NR1>

Description Sets and queries bits in the enable register for the Operation Status group. The enable register is a mask for enabling specific bits from the Operation Event register to set the OPER (operation summary) bit of the Status Byte register. STATus:PRESet clears all bits in the enable register.

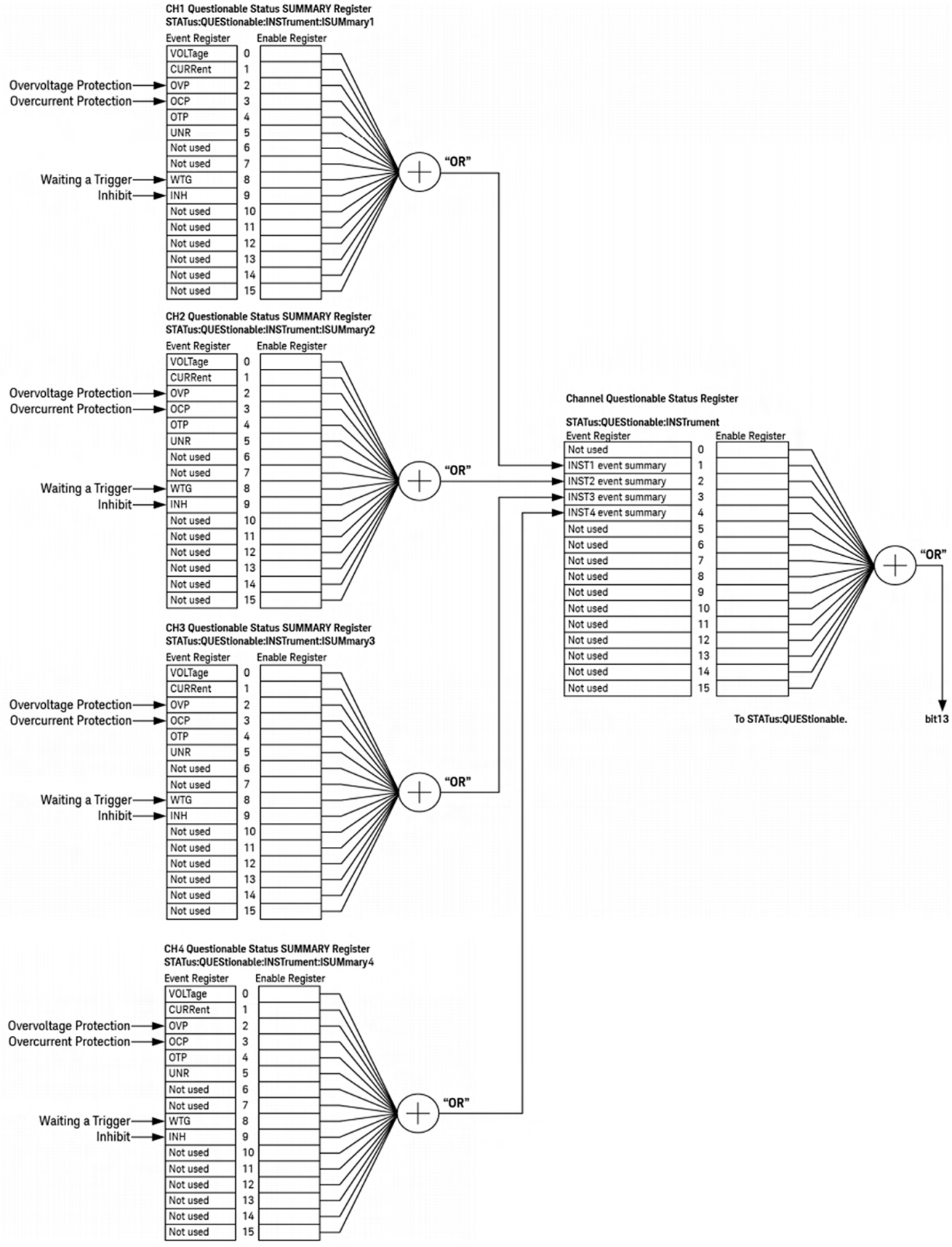
The query returns the bits in the enable register for the Operation Status group. The binary weighted value returned is in the **<NR1>** data type.

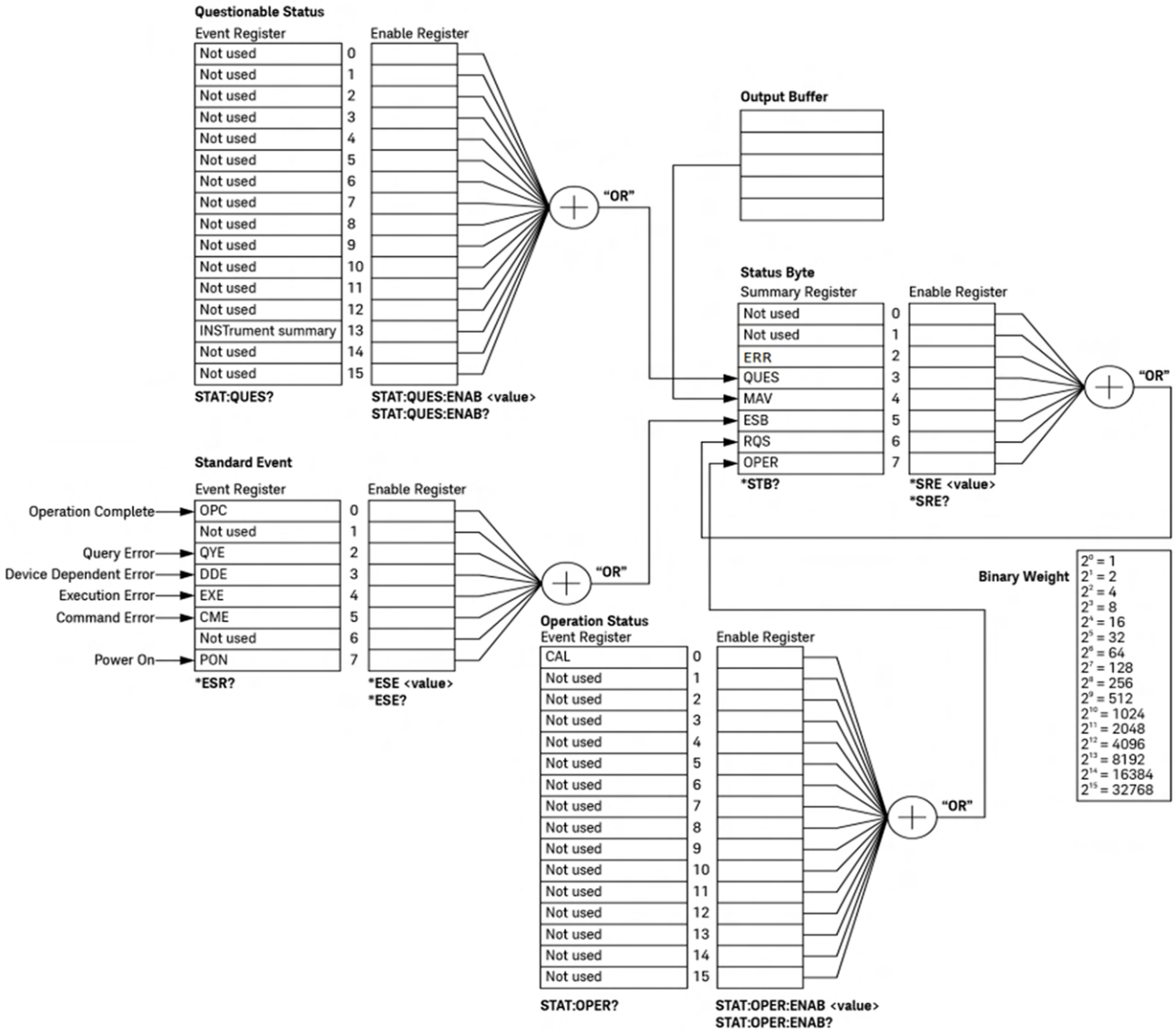
Syntax **Command** STATus:OPERation:ENABle **<NR1>**

Query STATus:OPERation:ENABle?

Parameters **<NR1>** := { 1 }

Example **Command** STAT:OPER:ENAB 2
Query STAT:OPER:ENAB?


CH1 Questionable Status SUMMARY Register
STATus:QUEStionable:INSTrument:ISUMmary1
Event Register
Enable Register
VOLTage 0
CURRent 1
Overvoltage Protection → OVP 2
Overcurrent Protection → OCP 3
OTP 4
UNR 5
Not used 6
Not used 7
Waiting a Trigger → WTG 8
Inhibit → INH 9
Not used 10
Not used 11
Not used 12
Not used 13
Not used 14
Not used 15
"OR"
CH2 Questionable Status SUMMARY Register
STATus:QUEStionable:INSTrument:ISUMmary2
Event Register
Enable Register
VOLTage 0
CURRent 1
Overvoltage Protection → OVP 2
Overcurrent Protection → OCP 3
OTP 4
UNR 5
Not used 6
Not used 7
Waiting a Trigger → WTG 8
Inhibit → INH 9
Not used 10
Not used 11
Not used 12
Not used 13
Not used 14
Not used 15
"OR"
CH3 Questionable Status SUMMARY Register
STATus:QUEStionable:INSTrument:ISUMmary3
Event Register
Enable Register
VOLTage 0
CURRent 1
Overvoltage Protection → OVP 2
Overcurrent Protection → OCP 3
OTP 4
UNR 5
Not used 6
Not used 7
Waiting a Trigger → WTG 8
Inhibit → INH 9
Not used 10
Not used 11
Not used 12
Not used 13
Not used 14
Not used 15
"OR"
CH4 Questionable Status SUMMARY Register
STATus:QUEStionable:INSTrument:ISUMmary4
Event Register
Enable Register
VOLTage 0
CURRent 1
Overvoltage Protection → OVP 2
Overcurrent Protection → OCP 3
OTP 4
UNR 5
Not used 6
Not used 7
Waiting a Trigger → WTG 8
Inhibit → INH 9
Not used 10
Not used 11
Not used 12
Not used 13
Not used 14
Not used 15
"OR"
Channel Questionable Status Register
STATus:QUEStionable:INSTrument
Event Register
Enable Register
Not used 0
INST1 event summary 1
INST2 event summary 2
INST3 event summary 3
INST4 event summary 4
Not used 5
Not used 6
Not used 7
Not used 8
Not used 9
Not used 10
Not used 11
Not used 12
Not used 13
Not used 14
Not used 15
"OR"
To STATus:QUEStionable.
bit13


Questionable Status
Event Register
Enable Register
Not used 0
Not used 1
Not used 2
Not used 3
Not used 4
Not used 5
Not used 6
Not used 7
Not used 8
Not used 9
Not used 10
Not used 11
Not used 12
INSTrument summary 13
Not used 14
Not used 15
STAT:QUES?
STAT:QUES:ENAB <value>
STAT:QUES:ENAB?
"OR"
Output Buffer
Status Byte
Summary Register
Enable Register
Not used 0
Not used 1
ERR 2
QUES 3
MAV 4
ESB 5
RQS 6
OPER 7
*STB?
*SRE <value>
*SRE?
"OR"
Standard Event
Event Register
Enable Register
Operation Complete
OPC 0
Not used 1
Query Error
QYE 2
Device Dependent Error
DDE 3
Execution Error
EXE 4
Command Error
CME 5
Not used 6
Power On
PON 7
*ESR?
*ESE <value>
*ESE?
"OR"
Binary Weight
2^0 = 1
2^1 = 2
2^2 = 4
2^3 = 8
2^4 = 16
2^5 = 32
2^6 = 64
2^7 = 128
2^8 = 256
2^9 = 512
2^10 = 1024
2^11 = 2048
2^12 = 4096
2^13 = 8192
2^14 = 16384
2^15 = 32768
Operation Status
Event Register
Enable Register
CAL 0
Not used 1
Not used 2
Not used 3
Not used 4
Not used 5
Not used 6
Not used 7
Not used 8
Not used 9
Not used 10
Not used 11
Not used 12
Not used 13
Not used 14
Not used 15
STAT:OPER?
STAT:OPER:ENAB <value>
STAT:OPER:ENAB?
"OR"

Bit	Bit Name	Decimal Value	Description
0	Voltage	1	Indicates the selected module is operating in constant voltage mode
1	Current	2	Indicates the selected module is operating in constant current mode
2	OVP	4	Indicates OVP (Over Voltage Protection) has been triggered on the selected module
3	OCP	8	Indicates OCP (Over Current Protection) has been triggered on the selected module
4	OTP	16	Indicates OTP (Over Temperature Protection) has been triggered on the selected module
5	UNR	32	Indicates that the selected module is unregulated; it is not regulated by CV or CC control loops
6	RNC	64	Indicates that remote sense is enabled but not connected in the selected module
7	WTG_DLOG	128	Indicates data logging is waiting for a trigger on the selected module
8	WTG	256	Indicates the selected module is waiting for a trigger
9	INH	512	Indicates the output of the selected module was disabled by an external INHibit signal

Table 17.1 Questionable Status SUMMARY Register

Bit	Bit Name	Decimal Value	Description
0	OPC	1	**Operation Complete**: All commands before and including *OPC have been executed
1	Not Used	2	Returns 0
2	QYE	4	**Query Error**: The output buffer was empty when attempting to read it, a new command was received before completing a previous query, or both the input and output buffers are full
3	DDE	8	**Device Specific Error**: A device-specific error, slef-test error, calibration error, or other device specific-error occurred
4	EXE	16	**Execution Error**: An execution error occurred
5	CME	23	**Command Error**: A command syntax error occurred
6	Not Used	64	Returns 0
7	PON	128	**Power On**: Power has been cycled since the last time the event register was read or cleared

Table 17.2 Standard Event Register

Bit	Bit Name	Decimal Value	Description
0	Not Used	1	Returns 0
1	Not Used	2	Returns 0
2	ERR	4	**Error Queue**: One or more errors are in the Error Queue. Errors are read and deleted using the **SYSTem:ERRor?**
3	QUES	8	**Questionable Status Summary**: One or more bits are set in the Questionable Data Register. Bits must be enabled using the command **STATus:QUEStionable:ENABle <NR1>**
4	MAV	16	**Message Available**: Data is avaialbel in the instrument's output buffer
5	ESB	23	**Event Status Summary**: One or more bits are set in the Standard Event Register. Bits must be enabled using the command ***ESE**
6	RQS	64	**Request Service Summary**: One or more bits are set in the Status Byte Register. Bits must be enabled using the command ***SRE**
7	OPER	128	**Operation Status Summary**: One or more bits are set in the Operation Status Register. Bits must be enabled using the command **STATus:OPERaton:ENABle <NR1>**

Table 17.3 Status Byte Register

System Subsystem

The SYSTem subsystem collects the functions that are not related to instrument performance. Examples include functions for performing general housekeeping and functions related to setting global configurations, such as TIME or SECurity.

18.1 SYSTem:BEEP[:IMMediate] <boolean> 82
18.2 SYSTem:BEEP:STATe <boolean> 82
18.3 SYSTem:CHANnel[:COUNt]? 83
18.4 SYSTem:CHANnel:MODel? 83
18.5 SYSTem:CHANnel:MODel:ALL? 83
18.6 SYSTem:CHANnel:SERIal? 83
18.7 SYSTem:CHANnel:SERIal:ALL? 84
18.8 SYSTem:CHANnel:VERSion? 84
18.9 SYSTem:CHANnel:VERSion:ALL? 84
18.10 SYSTem:COMMunicate:RLState <Characater> 85
18.11 SYSTem:COMMunicate:TCPip:CONTrol? 85
18.12 SYSTem:DATE <date> 86
18.13 SYSTem:ERRor? 86
18.14 SYSTem:LOCal 86
18.15 SYSTem:PERSona:MANufacturer? 87
18.16 SYSTem:PERSona:MODel? 87
18.17 SYSTem:REMote 87
18.18 SYSTem:RWLock 87
18.19 SYSTem:SECurity:IMMediate 88
18.20 SYSTem:VERSion? 88
18.21 SYSTem:TIME <time> 88

18.1 SYSTem:BEEP[:IMMediate] <boolean>

Description The command issues a single beep immediately.

Syntax **Command** SYSTem:BEEP[:IMMediate]

Parameters none

Example **Command** SYST:BEEP

18.2 SYSTem:BEEP:STATe <boolean>

Description The command enables/disables the beeper.

The query returns the state of the key press beeper. The data is returned in the **<boolean>** data type.

Syntax **Command** SYSTem:BEEP:STATe **<boolean>**

Query SYSTem:BEEP:STATe?

Parameters <**boolean**> := { 0 | 1 | ON | OFF }

Mode	Description
0	OFF
1	ON

Table 18.1 Key Press Beeper State

Example **Command** SYST:BEEP 1

Query SYST:BEEP?

18.3 SYSTem:CHANnel[:COUNt]?

Description The command returns the count of modules installed on the mainframe. (1 - 4)

Syntax **Query** SYSTem:CHANnel:COUNt?

Parameters none

Example **Query** SYST:CHAN?

18.4 SYSTem:CHANnel:MODel?

Description The command returns the model number of the active module. The active module is set using the **INSTrument[:SELect] <NR1>** command.

Syntax **Query** SYSTem:CHANnel:MODel?

Parameters none

Example **Query** SYST:CHAN:MOD?

18.5 SYSTem:CHANnel:MODel:ALL?

Description The command returns the model numbers of all installed modules, listing them sequentially from CH1 to CH4, with each number separated by a comma.

Syntax **Query** SYSTem:CHANnel:MODel:ALL?

Parameters none

Example **Query** SYST:CHAN:MOD:ALL?

18.6 SYSTem:CHANnel:SERIal?

Description The command returns the serial number of the active module. The active module is set using the **INSTrument[:SELect] <NR1>** command.

Syntax **Query** SYSTem:CHANnel:SERIal?

Parameters none

Example **Query** SYST:CHAN:SER?

18.7 SYSTem:CHANnel:SERIal:ALL?

Description The command returns the serial numbers of all installed modules, listing them sequentially from CH1 to CH4, with each number separated by a comma.

Syntax **Query** SYSTem:CHANnel:SERIal:ALL?

Parameters none

Example **Query** SYST:CHAN:SER:ALL?

18.8 SYSTem:CHANnel:VERSion?

Description The command returns the firmware version of the active module. The active module is set using the **INSTrument[:SELect] <NR1>** command.

Syntax **Query** SYSTem:CHANnel:VERSion?

Parameters none

Example **Query** SYST:CHAN:VER?

18.9 SYSTem:CHANnel:VERSion:ALL?

Description The command returns the firmware version installed on all modules, listing them sequentially from CH1 to CH4, with each number separated by a comma.

Syntax **Query** SYSTem:CHANnel:VERSion:ALL?

Parameters none

Example **Query** SYST:CHAN:VER:ALL?

18.10 SYSTem:COMMunicate:RLState <Characater>

Description The command configures the remote/local state of the instrument. Remote and Local do the same thing and are included for compatibility with other products. Both allow front panel control. RWLock disables the front panel keys. Then the instrument can only be controlled remotely.

The query returns the set communication state. The data is returned in the <Character> data type.

Syntax

Command SYSTem:COMMunicate:RLState <Characater> <Character>

Query SYSTem:COMMunicate:RLState?

Parameters <Character> := { LOCal | REMote | RWLock }

Mode	Description
Local	Sets the instrument to local mode, enables panel keys
Remote	Enables remote mode, front panel keys are disabled except for output control.
RWLock	Enables remote mode and disables all front panel keys.

Table 18.2 Key Press Beeper State

Example

Command SYST:COMM:RLS REM

Query SYST:COMM:RLS?

18.11 SYSTem:COMMunicate:TCPip:CONTrol?

Description The query returns the initial socket control connection port number. This connection is used to send and receive commands and queries. Unlike the data socket, which uses a fixed port number, the control socket port number varies and must be obtained using these queries. The data is returned in the **<NR1>** data type.

Syntax

Query SYSTem:COMMunicate:TCPip:CONTrol?

Parameters none

Example

Query SYST:COMM:TCP:CONT?

18.12 SYSTem:DATE <date>

Description The command configures the system's date.

The query returns the set date. The data is returned in the <date> data type.

Syntax **Command** SYSTem:DATE <date>

Query SYSTem:DATE?

Parameters <date> := { yyyy-mm-dd }

Example **Command** SYST:DATE 2024-6-6
Query SYST:DATE?

18.13 SYSTem:ERRor?

Description The query returns the system-defined error/event.

Syntax **Query** SYSTem:ERR?

Parameters none

Example **Query** SYST:ERR? Query the reported errors

18.14 SYSTem:LOCal

Description The command returns the instrument to local state, unlocking the controls of the instruments. This includes any front panel, keyboard, or other local interfaces that are locked in remote mode.

Syntax **Command** SYSTem:LOCal

Parameters none

Example **Command** SYST:LOC

18.15 SYSTem:PERSona:MANufacturer?

Description The command returns the instrument's manufacturer identity.

Syntax **Query** SYSTem:PERSona:MANufacturer?

Parameters none

Example **Query** SYST:PERS:MAN?

18.16 SYSTem:PERSona:MODel?

Description The command returns the mainframes's model number.

Syntax **Query** SYSTem:PERSona:MODel?

Parameters none

Example **Query** SYST:PERS:MOD?

18.17 SYSTem:REMote

Description The command sets the instrument to remote state, locking the controls of the instruments. This includes any front panel, keyboard, or other local interfaces.

Syntax **Command** SYSTem:REMote

Parameters none

Example **Command** SYST:REM

18.18 SYSTem:RWLock

Description The command sets the instrument to RWLock, locking all the controls of the instruments.

Syntax **Command** SYSTem:RWLock

Parameters none

Example **Command** SYST:RWL

18.19 SYSTem:SECurity:IMMediate

Description The command clears all user memory and reboots the instrument. This command is typically used to prepare the instrument for removal from a secure area. It sanitizes all user data by writing all zeros to flash memory and then performing a chip erase as per manufacturer's data sheet. Identification data (instrument firmware, model number, serial number, MAC address and calibration data) is not erased. After the data is cleared, the instrument is rebooted. This procedure is not recommended for use in routine applications because of the possibility of unintended loss of data.

Syntax **Command** SYSTem:SECurity:IMMediate

Parameters none

Example **Command** SYST:SEC:IMM

18.20 SYSTem:VERSion?

Description Returns the SCPI version that the instrument complies with. Cannot be determined from front panel.

Syntax **Query** SYSTem:VERSion?

Parameters none

Example **Query** SYST:VERS

18.21 SYSTem:TIME <time>

Description The command configures the system's time.

The query returns the set time. The data is returned in the <time> data type.

Syntax **Command** SYSTem:TIME <time>

Query SYSTem:TIME?

Parameters <time> := { hh:mm:ss }

Example **Command** SYST:TIME 09:16:30
Query SYST:TIME?

Trigger Subsystem

The trigger subsystem is used to synchronize device action(s) with events.

19.1 TRIGger:SEQuence:SOURce <NR1> 89

19.1 TRIGger:SEQuence:SOURce <NR1>

Description The command sets list's mode trigger source. This trigger source is seperate from the transient mode, and analog control triggers.

The query returns the set trigger mode for **List mode**. The returned value is in the <NR1> data type.

Syntax

Command TRIGger:SEQuence:SOURce <NR1>

Query TRIGger:SEQuence:SOURce?

Parameters <**Character**> := { IMM | EXT | BUS }

Mode	Description
IMM	Manual
EXT	Digital IO
BUS	Remote

Table 19.1 Trigger Source Parameters

Example

Command TRIG:SEQ:SOUR IMM

Query TRIG:SEQ:SOUR?