Programming Manual
9140 Series
Triple Output Multi-Range DC Power Supplies

bkprecision.com
## Contents

1 About Commands & Queries 3  
1.1 How They are Listed 3  
1.2 How They are Described 3  
1.3 When can they be used? 3  
1.4 Command Notation 3  

2 Common Command Introduction 4  
2.1 *CLS 5  
2.2 *ESE 5  
2.3 *ESR 5  
2.4 *IDN? 5  
2.5 *OPC 6  
2.6 *PSC 6  
2.7 RCL 6  
2.8 *SAV 7  
2.9 *RST 7  
2.10 *SRE 7  
2.11 *STB? 7  
2.12 *TRG 8  
2.13 *TST? 8  
2.14 *WAI 8  

3 Voltage Subsystem 9  
3.1 Measure Voltage 9  
3.2 Over Voltage Protection Limit 10  
3.3 Voltage Max. 10  
3.4 Voltage Min 10  
3.5 Voltage Protection Clear 10  
3.6 Voltage Protection State 10  
3.7 Voltage Sense 11  
3.8 Voltage Slew 11  
3.9 Voltage Source 11  

4 Current Subsystem 12  
4.1 Current Slope 12  
4.2 Current Source 12  
4.3 Measure Current 13  
4.4 Over Current Protection Clear 13  
4.5 Over Current Protection Limit 13  
4.6 Over Current Protection State 13  
4.7 Over Current Protection Tripped 13  

5 Power 14  
5.1 All Measurements 14  
5.2 Power Measurement 14  

6 Output Subsystem 15  
6.1 Operation Mode 15  
6.2 Output All 15  
6.3 Output State 16  
6.4 Output Timer Setting 16  
6.5 Output Timer State 16  
6.6 Power-On State 17  
6.7 Protection Clear 17
14 System Subsystem
14.1 Beeper
14.2 Date
14.3 Error
14.4 Error Next
14.5 Local Mode
14.6 LXI State
14.7 Memory Clear
14.8 Remote Mode
14.9 Remote Mode With Lock
14.10 Remote State
14.11 Single BEEP
14.12 Time
14.13 Version

15 Appendix
15.1 Output Configuration
15.2 List Edit
15.3 Running a List
15.4 Datalogger
15.5 Operation Modes
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.

### 1.1 How They are Listed

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

### 1.2 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.

### 1.3 When can they be used?

The commands and queries listed here can be used for 9140 Series Triple Output Multi-Range DC Power Supply.

### 1.4 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.
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.

<table>
<thead>
<tr>
<th>Short</th>
<th>Long Form</th>
<th>Subsystem</th>
<th>What Command/Query does</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CLS</td>
<td>*CLS</td>
<td>SYSTEM</td>
<td>Clears the instrument status byte by emptying the error queue and clearing all event registers. Also cancels any preceding *OPC command or query.</td>
</tr>
<tr>
<td>*ESE</td>
<td>*ESE</td>
<td>SYSTEM</td>
<td>Sets bits in the standard event status enable register.</td>
</tr>
<tr>
<td>*ESE?</td>
<td>*ESE?</td>
<td>SYSTEM</td>
<td>Returns the results of the standard event enable register. The register is cleared after reading it.</td>
</tr>
<tr>
<td>*ESR</td>
<td>*ESR</td>
<td>SYSTEM</td>
<td>Reads and clears the contents of the Event Status Register (ESR).</td>
</tr>
<tr>
<td>*IDN</td>
<td>*IDN</td>
<td>SYSTEM</td>
<td>Returns a string that uniquely identifies the instrument.</td>
</tr>
<tr>
<td>*OPC</td>
<td>*OPC</td>
<td>SYSTEM</td>
<td>Generates the OPC message in the standard event status register when all pending overlapped operations have been completed.</td>
</tr>
<tr>
<td>*OPC?</td>
<td>*OPC?</td>
<td>SYSTEM</td>
<td>Returns an ASCII &quot;+1&quot; when all pending overlapped operations have been completed.</td>
</tr>
<tr>
<td>*PSC?</td>
<td>*PSC?</td>
<td>SYSTEM</td>
<td>Gets or sets the OPC bit (0) in the Event Status Register (ESR).</td>
</tr>
<tr>
<td>*RCL</td>
<td>*RCL</td>
<td>SYSTEM</td>
<td>Recalls a saved instrument state.</td>
</tr>
<tr>
<td>*SAV</td>
<td>*SAV</td>
<td>SYSTEM</td>
<td>Save instrument state.</td>
</tr>
<tr>
<td>*RST</td>
<td>*RST</td>
<td>SYSTEM</td>
<td>Initiates a device reset.</td>
</tr>
<tr>
<td>*SRE</td>
<td>*SRE</td>
<td>SYSTEM</td>
<td>Set status byte enable register.</td>
</tr>
<tr>
<td>*SRE?</td>
<td>*SRE?</td>
<td>SYSTEM</td>
<td>Query status byte enable register.</td>
</tr>
<tr>
<td>*STB</td>
<td>*STB</td>
<td>SYSTEM</td>
<td>Query status byte.</td>
</tr>
<tr>
<td>*TRG</td>
<td>*TRG</td>
<td>SYSTEM</td>
<td>Generates an immediate trigger.</td>
</tr>
<tr>
<td>*TST?</td>
<td>*TST?</td>
<td>SYSTEM</td>
<td>Returns the result of the self-test.</td>
</tr>
<tr>
<td>*WAI</td>
<td>*WAI</td>
<td>SYSTEM</td>
<td>Prohibits the instrument from executing any new commands until all pending overlapped commands have been completed</td>
</tr>
</tbody>
</table>
2.1 *CLS

**Description**  The *CLS command clears the instrument status byte and all event registers. It also cancels any preceding *OPC commands and query.

**Example**  *CLS

2.2 *ESE

**Description**  The *ESE command sets the Standard Event Status Enable register (ESE). This command allows one or more events in the ESR register to be reflected in the ESB summary message bit (bit 5) of the STB register. The *ESE? query reads the contents of the ESE register.

**Command Syntax**  *ESE <value>

<value> := 0 to 255.

**Query Syntax**  *ESE?

**Query Response**  *ESE <value>

**Example**  *ESE?

2.3 *ESR

**Description**  The *ESR? query reads and clears the contents of the Event Status Register (ESR). The response represents the sum of the binary values of the register bits 0 to 7.

**Query Syntax**  *ESR?

**Query Response**  *ESR <value>

<value> := 0 to 255

**Example**  *ESR?

Return: 0

**Related Commands**  *CLS, *ESE

2.4 *IDN?

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

**Query Syntax**  *IDN?
**Response Syntax**  *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>:= Each product has its own number, the serial number can labeled product uniqueness.

<software version>:= A serial numbers about software version.

<hardware version>:= The hardware level field, should contain inSyntaxion about all separately revisable subsystems. This inSyntaxion can be contained in single or multiple revision codes.

**Example**  *IDN?

Returns: B&K Precision,9140,XXXXXXXX,0.25_1029A-0.15_0804A

### 2.5 *OPC

**Description**  The operation complete command causes the device to generate the operation complete message in the Standard Event Status Register, on completion of the selected device operation. The operation complete query places an ASCII character 1 in the output queue on completion of the selected device operation.

**Command Syntax**  *OPC

**Query Syntax**  *OPC?

**Example**  OUTP:STAT 1;*OPC

**Response Syntax**  1

### 2.6 *PSC

**Description**  Power on status clear command. This command is used to control the automatic power-on clearing of certain status functions.

**Command Syntax**  *PSC <state>

<state> := {0 | 1 }

**Query Syntax**  *PSC?

**Example**  *PSC

**Query Response**  *PSC?

Return: 0

### 2.7 RCL

**Description**  The *RCL command is used to restore the state of the device to that stored in the specified memory location.

**Command Syntax**  *RCL <memory address>

<memory address> := {0 to 9}
2.8 *SAV

Description The *SAV command saves instrument state.

Command Syntax *SAV

Example *SAV

2.9 *RST

Description The *RST command initiates a device reset. The *RST recalls the default setup.

Command Syntax *RST

Example *RST

2.10 *SRE

Description The *SRE command sets the Service Request Enable register (SRE). This command allows the user to specify which summary message bit(s) in the STB register will generate a service request.

A summary message bit is enabled by writing a ‘1’ into the corresponding bit location. Conversely, writing a ‘0’ into a given bit location prevents the associated event from generating a service request (SRQ). Clearing the SRE register disables SRQ interrupts.

The *SRE? query returns a value that, when converted to a binary number represents the bit settings of the SRE register. Note that bit 6 (MSS) cannot be set and it’s returned value is always zero.

Command Syntax *SRE <value>
<value> := 0 to 255

Command Syntax *SRE

Query syntax *SRE?

2.11 *STB?

Description The *STB? query reads the contents of the 488.2 defined status register (STB), and the Master Summary Status (MSS). The response represents the values of bits 0 to 5 and 7 of the Status Byte register and the MSS summary message.

The response to a *STB? query is identical to the response of a serial poll except that the MSS summary message appears in bit 6 in place of the RQS message.

Query Syntax *STB?

Query Response *STB <value>
<value> := 0 to 255

Example The following reads the status byte register:
*STB?
Related Commands  *CLS, *SRE

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Decimal Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 2</td>
<td>not used</td>
<td>not used</td>
<td>not used</td>
</tr>
<tr>
<td>3</td>
<td>QUES</td>
<td>8</td>
<td>Data Questionable Status Summary Bit.</td>
</tr>
<tr>
<td>4</td>
<td>MAV</td>
<td>16</td>
<td>Message available.</td>
</tr>
<tr>
<td>5</td>
<td>ESB</td>
<td>32</td>
<td>Standard Event Status Summary Bit.</td>
</tr>
<tr>
<td>6</td>
<td>RQS</td>
<td>64</td>
<td>Request Service (RQS) Summary Bit. A 1 in this bit position indicates that the signal generator has at least one reason to require service. This bit is also called the Master Summary Status bit (MSS). The individual bits in the Status Byte are individually ANDed with their corresponding service request enable register, then each individual bit value is ORed and input to this bit.</td>
</tr>
<tr>
<td>7</td>
<td>OPER</td>
<td>128</td>
<td>Operation Status Summary Bit.</td>
</tr>
</tbody>
</table>

2.12 *TRG

Description  The *TRG command generates an immediate trigger.

Command Syntax  *TRG

Example  This example resets the signal generator: *RST

2.13 *TST?

Description  The *TST? query performs an internal self-test and the response indicates whether the self-test has detected any errors. The self-test includes testing the hardware of all channels. Hardware failures are identified by a unique binary code in the returned <status> number. A “0” response indicates that no failures occurred.

Query Syntax  *TST?

Query Response  *TST <status>
<status> := 0 self-test successful

Example  TST?

Return: always returns 0

2.14 *WAI

Description  The *WAI (WAIT to continue) command, requires by the IEEE 488.2 standard, has no effect on the instrument, as the signal generator only starts processing a command when the previous command has been entirely executed.

Command Syntax  *WAI

Related Commands  *OPC
3.1 Measure Voltage

**Description**  Query the voltage measured at the specified output.

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

**Query Respond**  MEAS:SCAL:VOLTage:DC?

Returns: The output’s voltage.
## 3.2 Over Voltage Protection Limit

**Description** Set the OVP (Over Voltage Protection) Limit.

**Command Syntax** `SOURce:VOLTage:PROTection:LEVel:AMPLitude <value>`

**Query Syntax** `SOURce:VOLTage:PROTection:LEVel:AMPLitude?`

**Example** `SOUR:VOLT:PROT:LEV:AMPL 30`

**Query Respond**

```
SOUR:VOLT:PROT:LEV:AMPL?
```

Returns: 30.000

## 3.3 Voltage Max.

**Description** Set maximum voltage output.

**Command Syntax** `SOURce:VOLTage:MAXimum <value>`

**Query Syntax** `VOLTage:MAXimum?`

**Query Respond**

```
VOLT:MAX?
```

Returns: 60.600

## 3.4 Voltage Min.

**Description** Set minimum voltage output.

**Command Syntax** `SOURce:VOLTage:MINimum <value>`

**Query Syntax** `VOLTage:MINimum?`

**Query Respond**

```
VOLT:MIN?
```

Returns: 0.000

## 3.5 Voltage Protection Clear

**Description** Clear tripped OVP (Over Voltage Protection).

**Command Syntax** `SOURce:VOLTage:PROTection:CLEar`

## 3.6 Voltage Protection State

**Description** Query the state of OVP.

**Query Syntax** `SOURce:VOLTage:PROTection:TRIPped?`

**Query Respond**

```
SOURce:VOLTage:PROTection:TRIPped?
```

Returns: 0 if OVP is not tripped

1 if OVP is tripped
3.7 Voltage Sense

**Description**
Enable/disable the voltage sense.

**Command Syntax**
SOURce:VOLTage:SENSe <state>
<state> := { 0 or OFF := disable, 1 or ON := enable}

**Query Syntax**
SOURce:VOLTage:SENSe?

**Example**
SOURce:VOLTage:SENSe ON

**Query Respond**
SOURce:VOLTage:SENSe?

Returns: 1

3.8 Voltage Slew

**Description**
Set voltage slew.

**Command Syntax**
VOLT:SLOPe <value>

**Query Syntax**
VOLTage:SLOPe?

**Example**
VOLT:SLOP 3000

**Query Respond**
VOLT:SLOP?

Returns: 3000.000

3.9 Voltage Source

**Description**
Set the voltage output of the selected channel.

**Command Syntax**
SOURce:VOLTage:LEVel:IMMediate:AMPLitude <value>

**Query Syntax**
SOURce:VOLTage:LEVel:IMMediate:AMPLitude?

**Example**
SOUR:VOLT:LEV:IMM:AMPL 5

**Query Respond**
SOUR:VOLT:LEV:IMM:AMPL?

Returns: 5.000
4.1 Current Slope

Description: Set current slew.
Command Syntax: CURRent:SLOPe <value>
Query Syntax: CURRent:SLOPe?
Example: CURRent:SLOPe 250
Query Respond: CURRent:SLOPe?
Returns: 250.000

4.2 Current Source

Description: Set current output of the selected channel.
Command Syntax: SOURce:CURRent:LEVel:IMMediate:AMPLitude <value>
Query Syntax: SOURce:CURRent:LEVel:IMMediate:AMPLitude?
Example: SOUR:CURR:LEV:IMM:AMPL 2
Current Subsystem

**4.3 Measure Current**

*Description* Query the current measured at the specified output.

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

*Query Respond* MEAS:SCAL:CURR:DC?

Returns: The output’s current.

---

**4.4 Over Current Protection Clear**

*Description* Clear tripped OCP (Over Current Protection).

*Command Syntax* SOURce:CURRent:PROTection:CLEar

---

**4.5 Over Current Protection Limit**

*Description* Set OCP (Over Current Protection) Limit.

*Command Syntax* SOURce:CURRent:PROTection:LEVel <value>

*Query Syntax* SOURce:CURRent:PROTection:LEVel?

*Example* SOUR:CURR:PROT:LEV 8.8

*Query Respond* SOUR:CURR:PROT:LEV?

Returns: 8.800

---

**4.6 Over Current Protection State**

*Description* Enable/disable OCP (Over Current Protection).

*Command Syntax* SOURce:CURRent:PROTection:STATe <state>

<state> := { 0 or OFF := disable, 1 or ON := enable}

*Query Syntax* SOURce:CURRent:PROTection:STATe?

*Example* SOUR:CURR:PROT:STAT OFF

*Query Respond* SOUR:CURR:PROT:STAT?

Returns: 0

---

**4.7 Over Current Protection Tripped**

*Description* Query the state of OCP (Over Current Protection).

*Query Syntax* SOURce:CURRent:PROTection:TRIPped?

*Query Respond* SOUR:CURR:PROT:TRIP?

Returns: 0 if OVP is not tripped

1 if OVP is tripped
5.1 All Measurements

- **Description**: Query voltage, current, and power.
- **Query Format**: `MEASure:ALL?`
- **Example**: `MEAS:ALL?`
- **Query Respond**: Returns: 10,2,20

5.2 Power Measurement

- **Description**: Query power
- **Query Format**: `MEASure:POWer?`
- **Example**: `MEAS:POW?`
- **Query Respond**: Returns: 20
6.1 Operation Mode

Description: Set the operation mode to either OFF|PARA|SERI|TRAC.

Command Syntax: OUTPut:PAIR <mode>

- <mode> := {OFF |PARA2 |PARA3 |SERI2 |SERI3 |TRAC2 |TRAC3}
- OFF := Normal Mode
- PARA <2|3> := { Parallel Mode ; 2 := CH1+CH2 ; 3 := All Channels}
- SERI <2|3> := { Parallel Mode ; 2 := CH1+CH2 ; 3 := All Channels}
- TRAC <2|3> := {Parallel Mode ; 2 := CH1+CH2 ; 3 := All Channels}

Query Syntax: OUTPut:PAIR?

Example: OUTP:PAIR PARA3

Query Respond: OUTP:PAIR?

Returns: PARA3

6.2 Output All

Description: Enable/disable all channels simultaneously.

Command Syntax: OUTPut:ALL <state>

- <state> := {1 |0 or On |Off}

Query Syntax: OUTPut:ALL?

Example: OUTP:ALL 1

Query Respond: OUTP:ALL?

Returns: 1
6.3 Output State

**Description**
Set the output state on the selected channel ON|OFF.

**Command Syntax**
```
OUTPut:STATe <state>
```

\(<state> := \{0 | 1 | OFF | ON\}\)

**Query Syntax**
```
OUTPut:STATe?
```

**Example**
```
OUTP:STAT ON
```

**Query Respond**
```
OUTP:STAT?
```

Returns: 1

6.4 Output Timer Setting

**Description**
Set the run time of the output timer.

**Command Syntax**
```
OUTPut:TIMer:COUNt <time>
```

\(<time> := \{0 \text{ to } 00:50:49\}\)

**Query Syntax**
```
OUTPut:TIMer:COUNt?
```

**Example**
```
OUTP:TIM:COUN 10,00,00
```

**Query Respond**
```
OUTP:TIM:COUN?
```

Returns: 0,0,0

6.5 Output Timer State

**Description**
Set the state of the output timer.

**Command Syntax**
```
OUTPut:TIMer <state>
```

\(<state> := \{0 | 1 | OFF | ON\}\)

**Query Syntax**
```
OUTPut:TIMer?
```

**Example**
```
OUTP:TIM ON
```

**Query Respond**
```
OUTP:TIM?
```

Returns: 0 if off and 1 if on.
6.6 Power-On State

**Description**  Set power-on state.

**Command Syntax**  `OUTPut:PON:STATe <state>,<user>`

\(<state> := \{\text{OFF, LAST, \text{<USER,@ programmed user>}}\}\)

**Query Syntax**  `OUTPut:PON:STATe?`

**Example**  `OUT:PON:STAT USER,2`

**Query Respond**  `OUT:PON:STAT?`

Returns: `USER,2`

6.7 Protection Clear

**Description**  Clear all triggered protections in all channels.

**Command Syntax**  `OUTPut:PROTection:CLear`

**Example**  `OUTP:PROT:CLE`
7.1 Inhibit Mode

**Description**  Set inhibit mode to either LATChing|LIVE|OFF.

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

*<mode>* := {LATChing |LIVE |OFF}

**Query Syntax**  OUTPut:INHibit:MODE?

**Example**  OUTP:INH:MODE LIVE

**Query Respond**  OUTP:INH:MODE?

Returns: LIVE

**Note:**
Inhibit mode must be enabled. See section 7.4

---

7.2 Digital Input State

**Description**  Query the state of the digital control port pins. Each of the three pins can be configured as digital input only.

**Query Syntax**  SOURce:DIgital:INPut:DATA?

**Query Respond**  SOUR:DIg:INP:DATA?

Return: 0 to 7.
7.3 Digital Output

**Description**
A Digital Control Port consisting of three I/O pins is provided to access various control functions. Data is programmed as followed:
Pin 1 is most significant bit, while pin 3 is the least significant bit.

**Command Syntax**
SOURce:DIGital:OUTPut:DATA <Data>  <Data> := {0 to 7}

**Query Syntax**
SOURce:DIGital:OUTPut:DATA?

**Example**
To send a binary weighted value configure pins 1 through 3 as 111 (7) if all polarities are set to pos. The polarity will affect the output set.
SOUR:DIG:OUTP:DATA 7

**Query Respond**
SOUR:DIG:OUTP:DATA?
Returns: 7

7.4 Digital Pin Functions

**Description**
Set the function of the digital port pins.

**Command Syntax**
SOURce:DIGital:PIN<pin number>:FUNCtion <function>
<pin number> := { 1, 2, or 3}

<function> := { NONE | DOUT | DINP | TOUT | TINP | FAUL | INH }

**Note:**
Inhibit IN (INH) is only available for pin 3.

**Query Syntax**
SOURce:DIGital:PIN<pin number>:FUNCtion?

**Example**
SOUR:DIG:PIN1:FUNC NONE

**Query Respond**
SOUR:DIG:PIN1:FUNC?
Returns: NONE

7.5 Digital Pin Polarity

**Description**
Set the polarity of the digital port pins.

**Command Syntax**
SOURce:DIGital:PIN<pin number>:POLarity <polarity>
<pin number> := { 1|2|3}

<polarity> := { POS | NEG}

**Query Syntax**
SOURce:DIGital:PIN<pin number>:POLarity?

**Example**
SOUR:DIG:PIN1:POL POS

**Query Respond**
SOUR:DIG:PIN1:POL?
Returns: POS
8.1 After List

Description  Set the list terminate state.

Command Syntax  SOURce:LIST:TERMinate:LAST <state>

<state> := {1 & ON := Last | 0 & OFF := DC}

Query Syntax  SOURce:LIST:TERMinate:LAST?

Example  SOUR:LIST:TERM:LAST ON

Query Respond  SOUR:TERM:LAST?

Returns: 1

8.2 BOST

Description  Specifies which list steps generate a trigger signal at the beginning of the step.

Command Syntax  SOURce:LIST:TOUTput:BOSTep:DATA <bool>

<voltage> := {0 | 1}

Query Syntax  SOURce:LIST:TOUTput:BOSTep:DATA?

Example  SOUR:LIST:TOUT:BOST:DATA 1

Query Respond  SOUR:LIST:BOST:TOUT:DATA?

Returns: 1
### 8.3 EOST

**Description**  
Specifies which list steps generate a trigger signal at the end of the step.

**Command Syntax**  
```
SOURce:LIST:TOUTput:EOSTep:DATA <bool>
```

\(<voltage> := \{0 | 1\}\)

**Query Syntax**  
```
SOURce:LIST:TOUTput:EOSTep:DATA?
```

**Example**  
```
SOUR:LIST:TOUT:EOST:DATA 1
```

**Query Respond**  
```
SOUR:LIST:TOUT:EOST:DATA?
```

Returns: 1

### 8.4 List Abort

**Description**  
Disable list function.

**Command Syntax**  
```
OUTPut:MODE FIXed
```

**Example**  
```
OUTP:MODE FIX
```

### 8.5 List Clear

**Description**  
Clears all set function of the selected list.

**Command Syntax**  
```
LIST:CLEar <list>
```

\(<list> = \{1 to 10\}\)

**Example**  
```
SOUR:LIST:NUMB 1;:LIST:CLE;:SOUR:LIST:SAVE
```

All parameters of the selected list one will be deleted. Step 1 will remain with the step default values.

Default Values: 0 V, .015 A, BOST and EOST disabled, Dwell= .1 s

**Note:**

Before clearing all parameters enter **List Edit Mode** by sending the command SOURce:LIST:NUMBer <list>. Where \(<list>\) is the list to be configured. Once the command to clear has been sent save the changes using the command SOURC:LIST:SAVE.

### 8.6 List Count

**Description**  
Set the list repeat.

**Command Syntax**  
```
SOURce:LIST:COUNt <count>
```

\(<count> = \{1 to 100000\}\)

**Query Syntax**  
```
SOURce:LIST:COUNt?
```

**Example**  
```
SOUR:LIST:COUN 10
```
8.7 List Delete

**Description**
Deletes the specified step and all steps that follow it in the selected list.

**Command Syntax**
LIST:DELETE <step>

<step> = {1 to 100}

**Example**
SOUR:LIST:NUMB 1; LIST:DEL 10; SOUR:LIST:SAVE
- In a list containing 100 steps the command will delete the steps 10 to 100.
- In a list containing 30 steps the command will delete the steps 10 to 30.

**Note:**
Before deleting any steps enter **List Edit Mode** by sending the command SOURce:LIST:NUMBer <list>. Where <list> is the list to be configured. Once the command to delete the desired steps has been sent save the changes using the command SOURC:LIST:SAVE. **This command does not delete individual steps.**

8.8 List Next

**Description**
Set the next list to be executed the current list elapses. You can make a list run infinitely by setting list to the current list.

**Command Syntax**
SOURce:LIST:NEXT <list>

<list> := {0 to 10}

0 := no list will run after current list elapses 1 to 10 := List that will run after current list elapses.

**Query Syntax**
SOURce:LIST:NEXT?

**Example**
SOUR:LIST:NEXT 1

**Query Respond**
SOUR:LIST:NEXT?
Returns: 1

8.9 List Number

**Description**
Enable list state and assign a list from the instrument’s memory to the channel.

**Command Syntax**
SOURce:LIST:NUMBer <number>

<number> := {1 to 10}

**Query Syntax**
SOURce:LIST:NUMBer?

**Example**
SOUR:LIST:NUMB 2
8.10 List Output

**Description**
Assign a programmable list to selected channel.

**Command Syntax**
OUTPut:MODE LIST,<list>

<list> = {1 to 10}

**Example**
OUTP:MODE LIST,2

8.11 List Pace

**Description**
Set the pace of the list.

**Command Syntax**
SOURce:LIST:STEP <state>

<state> := {1 & ON := Trigger | 0 & OFF := Dwell}

**Query Syntax**
SOURce:LIST:STEP?

**Example**
SOUR:LIST:STEP ON

**Query Respond**
SOUR:LIST:STEP?
Returns: 1

8.12 List Start/Trigger Initiate

**Description**
Start the list on the specified channel. If continuous trigger is disabled INIT <channel> must be for the unit to read the *TRG command.

**Command Syntax**
INITiate[:IMMediate] <channel>

<channel> := {0|1|2}

0 := channel 1
1 := channel 2
2 := channel 3

**Example**
INIT 0
8.13 List Start Continuous Trigger

**Description**
Enable/disable continuous triggers.

**Command Syntax**
INITiate:CONTinuous <channel>,<state>

- `<channel>` := {0|1|2}
  - 0 := channel 1
  - 1 := channel 2
  - 2 := channel 3

- `<state>` := {0 | 1}

**Example**
INIT:CONT 0,1

8.14 List Status

**Description**
Query the list state of the selected channel.

**Query Syntax**
LIST:STATe?

**Example**
LIST:STAT?

**Query Respond**
LIST:STAT?
Returns: 1 if list is running 0 if list is not running.

8.15 List Trigger Source

**Description**
Set state of steps enable |disable.

**Command Syntax**
TRIGger:SEQuence:SOURce <source>

- `<source>` := {IMMediate|BUS|EXTernal}
  - IMM := Manual control
  - BUS := Remote control
  - EXT := Digital I/O control

**Query Syntax**
TRIGger:SEQuence:SOURce?

**Example**
TRIG:SEQ:SOUR BUS

**Query Respond**
TRIG:SEQ:SOUR?
Returns: BUS
8.16 Save List

**Description**  
Save data set in temporary memory to the selected list. **To save the changes no list can be initiated.**

**Command Syntax**  
SOURce:LIST:SAVE

**Example**  
SOUR:LIST:SAVE

8.17 Step Current

**Description**  
Set the current for each list step.

**Command Syntax**  
SOURce:LIST:CURRent:LEVel <current>

<current> := {0 to 24 Amps}

Current output range will vary depending on model and mode.

**Query Syntax**  
SOURce:LIST:CURRent:LEVel?

**Example**  
SOUR:LIST:CURR:LEV 2

**Query Respond**  
SOUR:LIST:CURR:LEV?

Returns: 2

8.18 Step Voltage

**Description**  
Set the voltage for each list step.

**Command Syntax**  
SOURce:LIST:VOLTage:LEVel <voltage>

<voltage> := {0 to 96 V}

Voltage output range will vary depending on model and mode.

**Query Syntax**  
SOURce:LIST:VOLTage:LEVel?

**Example**  
SOUR:LIST:VOLT:LEV 5

**Query Respond**  
SOUR:LIST:VOLT:LEV?

Returns: 5
8.19 List Edit

**Description**  
Editing a list requires the commands to be sent in a sequence. If the commands are not sent in the correct sequence the unit will beep indicating it does not recognize the command.

**Additional Information**  
A list can be broken down into 3 parts: List Setup, List Parameters, and Step Parameters. List setup is independent of list parameter and step parameters.

**List Setup**  
List setup includes the following parameters:

<table>
<thead>
<tr>
<th>List State</th>
<th>List Number</th>
<th>List Pace</th>
<th>Trigger Source</th>
<th>After List</th>
</tr>
</thead>
</table>

These parameters are independent and can be set at any point.

**List and Step Parameters**  
The list and step parameters are codependent and must be sent in a sequence for the unit to recognize the commands.

**Note:**

While editing a list’s parameters new changes will be stored in temporary memory. SOURC:LIST:SAVE must be used to save the changes to the internal memory.

**ABORT all list initiated in any channels before you begin the sequence.**

---

**Sequence**

1. Enter edit mode of a specified list. Before entering edit mode abort all running list. Only one list can be edited at a time.
   - SOUR:LIST:NUMB <list>; list := {1 to 10}

2. Set the list parameters:
   - SOUR:LIST:COUN <repeat>  
     repeat := {1 to 99999}
   - SOUR:LIST:NEXT <next>  
     next := {0|1 to 10} 0 := off

3. To edit a step enter step edit mode using:
   - SOURce:LIST:STEP:NUMBER <number>;number := {1 to 100}

4. Edit the step’s parameters (ranges may vary depending on model and output mode)
   - SOURce:LIST:VOLTage:LEVel <voltage>  
     voltage := {0 to 180 V}
   - SOURce:LIST:CURRent:LEVel <current>  
     current := {0 to 24 A}
   - SOURce:LIST:TOUTput:BOSTep:DATA <state>  
     state := {1|0|On|Off}
   - SOURce:LIST:TOUTput:EOSTep:DATA <state>  
     state := {1|0|On|Off}
   - SOURce:LIST:DWELl <time>  
     time := {0.|1 to 9999} 0 := terminate step

5. Save any changes made using the command:
   - SOURce:LIST:SAVE
**Examples**

To only edit the **List Parameters**:

```
```

To edit both the **List and Step Parameters**:

```
SOUR:LIST:NUMB 2; SOUR:LIST:COUN 1; SOUR:LIST:NEXT 5;
SOUR:LIST:STEP:NUMB 1
SOUR:LIST:CURR:LEV 1; SOUR:LIST:VOLT:LEV 1; SOUR:LIST:DWELI
```

**Note:**

When editing the step parameters all 5 parameters must be set even if no change is being made to one of them. If not all parameters are configured the unit will not accept a command to edit another step until all parameters have been set.

**8.20 Running a List**

**Description** After setting a list's parameters, run the list by following one of the sequence below. Example 1 demonstrated how to run a list with pace set to **Trigger**. Example 2 demonstrates how to run a list with pace set to **Dwell**.

**Examples 1**

<table>
<thead>
<tr>
<th>INST 0</th>
<th>select a channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTP:MODE LIST,1</td>
<td>enable and assign list</td>
</tr>
<tr>
<td>OUTP 1</td>
<td>turn output on</td>
</tr>
<tr>
<td>INIT 1</td>
<td>start list on selected channel</td>
</tr>
<tr>
<td>*TRG</td>
<td>send a trigger</td>
</tr>
<tr>
<td>INIT 1</td>
<td>abort list on selected channel</td>
</tr>
<tr>
<td>*TRG</td>
<td>exit list mode</td>
</tr>
</tbody>
</table>

**Examples 2**

<table>
<thead>
<tr>
<th>INST 0</th>
<th>select a channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTP:MODE LIST,1</td>
<td>enable and assign list</td>
</tr>
<tr>
<td>OUTP 1</td>
<td>turn output on</td>
</tr>
<tr>
<td>INIT 0</td>
<td>start list on selected channel</td>
</tr>
<tr>
<td>ABOR 0</td>
<td>abort list on selected channel</td>
</tr>
<tr>
<td>OUTP:MODE FIX</td>
<td>exit list mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INST 0</th>
<th>select a channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTP:MODE LIST,1</td>
<td>enable and assign list</td>
</tr>
<tr>
<td>OUTP 1</td>
<td>turn output on</td>
</tr>
<tr>
<td>INIT 0</td>
<td>start list on selected channel</td>
</tr>
<tr>
<td>ABOR 0</td>
<td>abort list on selected channel</td>
</tr>
<tr>
<td>OUTP:MODE FIX</td>
<td>exit list mode</td>
</tr>
</tbody>
</table>

**Dwell**
9.1 Display Window

**Description**  Lock the front-panel keys.

**Command Syntax**  
(DISPlay:WNDow:STATe <state>)

<state> := {ON|OFF}

**Query Syntax**  
(DISPlay:WNDow:STATe?)

**Example**  
DISP:WND:STAT ON

**Query Respond**  
DISP:WND:STAT?

Returns: 1
10.1 Channel Select

Description: Select the channel to be programmed.

Command Syntax: INSTrument:SELect <channel>
   <channel> = {0 | 1 | 2}

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

Query Format: INStrument:SELect?
Example: INST:SEL 2
Query Respond: INST:SEL?
   Returns: CH 3

10.2 Apply

Description: Set and query the voltage and current of the selected channel.

Command Syntax: APPLy <voltage>,<current>

Query Syntax: APPLy?
Example: APPL 10,1
Query Respond: APPL?
   Returns: 10,1
11.1 Abort

**Description**  
Clear any pending delayed trigger and returns the trigger system to idle.

**Command Syntax**  
ABORt <channel>

<channel> := {0|1|2}

**Note:**  
The programmable outputs begin at 0. Therefore, to program channel 1 select 0, to program channel 2 select 1, and to program channel 3 select 2.
12.1 DataLog Abort

**Description** Stop the present data logging session.

**Command Syntax** `ABORt:DLOG`

12.2 Data Logger Initiation

**Description** Toggle the data logging session ON | OFF. Data recording does not begin until Data Logger Trigger signal is received.

**Command Syntax** `INITiate:IMMediate:DLOG`

**NOTE:** A flash drive must be connected to use this command.

**Example** `INIT:IMM:DLOG`

12.3 Data Logger Record Trigger

**Description** Sends an immediate trigger signal to start the data logger recording.

**Command Syntax** `TRIGger:DLOG:IMMediate`

**NOTE:** A flash drive must be connected to use this command.

**Example** `TRIG:DLOG:IMM`
## 12.4 Data Logger Trigger Source

**Description**
Select the trigger source for the data logger.

**Command Syntax**
`TRIgger:DLOG:SOURce <source>`

\(<source> = \{BUS|EXTernal|IMMediate\}`

**Query Syntax**
`TRIgger:DLOG:SOURce?`

**Example**
`TRIG:DLOG:SOUR BUS`

**Query Respond**
`TRIG:DLOG:SOUR?`
Returns: BUS

## 12.5 Sample Interval

**Description**
Sets the sample period consisting of the entered value in seconds.

**Command Syntax**
`SENSe:DLOG:FUNCtion:TINTerval <time>`

\(<time> := \{.2 \text{ to } 300 \text{ seconds}\}`

**Query Syntax**
`SENSe:DLOG:FUNCtion:TINTerval ?`

**Example**
`SENS:DLOG:FUNC:TINT 1`

**Query Respond**
`SENS:DLOG:FUNC:TINT?`
Returns: 1

## 12.6 Sample Period

**Description**
Set the sample period in seconds.

**Command Syntax**
`SENSe:DLOG:FUNCtion:PER <time>`

\(<time> := \{.2 \text{ to } 300 \text{ seconds}\}`

**Query Syntax**
`SENSe:DLOG:FUNCtion:PER?`

**Example**
`SENS:DLOG:FUNC:PER 1`

**Query Respond**
`SENS:DLOG:FUNC:PER?`
Returns: 1
13.1 Preset Status

Description  Clears all bits in the enable register.

Command Syntax  STATus:PRESet

Example  STAT:PRES

13.2 Questionable Instrument Enable

Description  Set the questionable status enable register.

Command Syntax  STATus:QUEStionable:INSTrunent:ENABle <bit value>

Query Format  STATus:QUEStionable:INSTrunent:ENABle?

Example  STAT:QUES:INST:ENAB 13

Query Respond  STAT:QUES:INST:ENAB?

Returns: 13

13.3 Questionable Instrument Event

Description  Query questionable status event register and clears the register.

Query Syntax  STATus:QUEStionable:INSTrunent:EVENt?

Query Respond  STAT:QUES:INST:EVEN?

Returns: 0
13.4 Questionable Isummary

Description
Query the questionable instrument isummary event register and clears the register.

Query Syntax
STATus:QUEStionable:ISUMmary<Channel>:EVENt?

Query Respond
STAT:QUES:ISUM2:EVEN?

Returns: See table Questionable Status Summary in 13.9

13.5 Questionable Isummary Condition

Description
Query the CV or CC condition of the specified output.

Query Syntax
STATus:QUEStionable:ISUMmary:CONDition?

Query Respond
STAT:QUES:ISUM:COND?

Returns: 0

13.6 Questionable Isummary Enable

Description
Set the value of the questionable instrument isummary enable register for a specific output.

Command Syntax
STATus:QUEStionable:ISUMmary:ENABle <bit value>

Query Format
STATus:QUEStionable:ISUMmary:ENABle?

Example
STAT:QUES:ISUM:ENAB 0

Query Respond
STAT:QUES:ISUM:ENAB?

Returns: 0

13.7 Questionable Status

Description
Query questionable register.

Query Syntax
STATus:QUEStionable:EVENt?

Example
STAT:QUES:EVEN?

Returns:

13.8 Questionable Status Enable

Description
Set questionable status enable register.

Command Syntax
STATus:QUEStionable:ENABle <bit value>

Query Format
STATus:QUEStionable:ENABle?

Example
STAT:QUES:ENABle 0

Query Respond
STAT:QUES:ENAB?

Returns: 0
13.9 Status Diagrams

**Description**  The following diagrams shows the capability of the status reporting. The status data structure-register model is represented in the boxes. The logical summing is shown with the circle containing a +.

### Questionable Status Summary

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Decimal Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>VOLTage</td>
<td>1</td>
<td>Bit is 1 when power supply is operating as voltage source.</td>
</tr>
<tr>
<td>1</td>
<td>CURRent</td>
<td>2</td>
<td>Bit is 1 when power supply is operating as current source.</td>
</tr>
<tr>
<td>2</td>
<td>OVP</td>
<td>4</td>
<td>Bit is 1 when OVP (Over Voltage Protection) occurs.</td>
</tr>
<tr>
<td>3</td>
<td>OCP</td>
<td>8</td>
<td>Bit is 1 when OVP (Over Current Protection) occurs.</td>
</tr>
<tr>
<td>4</td>
<td>OTP</td>
<td>16</td>
<td>Bit is 1 when OTP (Over Temperature Protection) occurs.</td>
</tr>
<tr>
<td>5</td>
<td>UNR</td>
<td>32</td>
<td>Bit is 1 when output is unregulated.</td>
</tr>
<tr>
<td>6</td>
<td>RNC</td>
<td>64</td>
<td>Bit is 1 when remote sense is not connected.</td>
</tr>
<tr>
<td>7</td>
<td>WTG_DLOG</td>
<td>128</td>
<td>Bit is 1 when DLOG is waiting for a trigger.</td>
</tr>
<tr>
<td>8</td>
<td>WTG</td>
<td>256</td>
<td>Bit is 1 when instrument is waiting for a trigger.</td>
</tr>
<tr>
<td>9</td>
<td>INH</td>
<td>512</td>
<td>Bit is 1 when inhibit occurs.</td>
</tr>
</tbody>
</table>

### Standard Event

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Decimal Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OPC</td>
<td>1</td>
<td>Bit is 1 when all commands before and including *OPC have been executed.</td>
</tr>
<tr>
<td>1</td>
<td>not used</td>
<td>not used</td>
<td>not used</td>
</tr>
<tr>
<td>2</td>
<td>QYE</td>
<td>4</td>
<td>Bit is 1 when: the instrument tried to read the output buffer but it was empty, a new command was received before a previous query had been read, or both the input and output buffers are full.</td>
</tr>
<tr>
<td>3</td>
<td>DDE</td>
<td>8</td>
<td>Bit is 1 when a device-specific error occurred.</td>
</tr>
<tr>
<td>4</td>
<td>EXE</td>
<td>16</td>
<td>Bit is 1 when an execution error occurred.</td>
</tr>
<tr>
<td>5</td>
<td>CME</td>
<td>32</td>
<td>Bit is 1 when a command syntax error occurred.</td>
</tr>
<tr>
<td>6</td>
<td>not used</td>
<td>64</td>
<td>not used</td>
</tr>
<tr>
<td>7</td>
<td>PON</td>
<td>128</td>
<td>Bit is 1 when power has been cycled since the last time the event register was read or cleared.</td>
</tr>
</tbody>
</table>

### Status Byte

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Decimal Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 2</td>
<td>not used</td>
<td>not used</td>
<td>not used</td>
</tr>
<tr>
<td>3</td>
<td>QUES</td>
<td>8</td>
<td>Bit is 1 when one or more bits are set in the Questionable Data Register. Bits must be enabled, see STATus:QUESTionable:ENABle.</td>
</tr>
<tr>
<td>4</td>
<td>MAV</td>
<td>16</td>
<td>Bit is 1 when data is available in the instrument’s output buffer.</td>
</tr>
<tr>
<td>5</td>
<td>ESB</td>
<td>32</td>
<td>Bit is 1 when one or more bits are set in the status byte register and may generates a service request.</td>
</tr>
<tr>
<td>6</td>
<td>RQS</td>
<td>64</td>
<td>Request Service (RQS) Summary Bit: A 1 in this bit position indicates that the signal generator has at least one reason to require service. This bit is also called the Master Summary Status bit (MSS). The individual bits in the Status Byte are individually ANDed with their corresponding service request enable register, then each individual bit value is ORed and input to this bit.</td>
</tr>
<tr>
<td>7</td>
<td>OPER</td>
<td>128</td>
<td>Bit is 1 when one or more bits are set in the operation status register.</td>
</tr>
</tbody>
</table>
14.1 Beeper

**Description**  Set the beeper state.

**Command Syntax**  
SYSTem:BEEPer:STATe <state>

<state> = {ON | OFF | 1 | 0}

**Query Syntax**  
SYSTem:BEEPer:STATe?

**Example**  
SYST:BEEP:STAT 0

**Query Respond**  
SYST:BEEP:STAT?

Returns: 0

14.2 Date

**Description**  Set the date of the power supply’s real-time clock.

**Command Syntax**  
SYSTem:DATE <YY>,<MM>,<DD>

**Query Syntax**  
SYSTem:DATE?

**Example**  
SYST:DATE 21,2,17

**Query Respond**  
SYST:DATE?

Returns: 2020,11,4
### 14.3 Error

**Description**  Query and clear the first error from the error queue (FIFO).

**Query Syntax**  SYSTem:ERRor?

**Query Respond**  SYST:ERR?

The following is one of the possible errors:
-109, Missing parameter

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0</td>
<td>No error</td>
</tr>
<tr>
<td>-102</td>
<td>Syntax error</td>
</tr>
<tr>
<td>-103</td>
<td>Invalid separator</td>
</tr>
<tr>
<td>-104</td>
<td>Data type error</td>
</tr>
<tr>
<td>-105</td>
<td>GET not allowed</td>
</tr>
<tr>
<td>-108</td>
<td>Parameter not allowed</td>
</tr>
<tr>
<td>-109</td>
<td>Missing parameter</td>
</tr>
<tr>
<td>-110</td>
<td>Command header error</td>
</tr>
<tr>
<td>-111</td>
<td>Header separate error</td>
</tr>
<tr>
<td>-113</td>
<td>Undefined header</td>
</tr>
<tr>
<td>-131</td>
<td>Invalid suffix</td>
</tr>
<tr>
<td>-238</td>
<td>Suffix not allowed</td>
</tr>
<tr>
<td>-203</td>
<td>Command protected</td>
</tr>
<tr>
<td>-221</td>
<td>Settings conflict</td>
</tr>
<tr>
<td>-222</td>
<td>Data out of range</td>
</tr>
<tr>
<td>-223</td>
<td>Too much data</td>
</tr>
<tr>
<td>-240</td>
<td>Hardware error</td>
</tr>
<tr>
<td>-350</td>
<td>Error queue overflow</td>
</tr>
</tbody>
</table>

**Table 14.1**  Error Codes

### 14.4 Error Next

**Description**  Query and clear the first error from the error queue (FIFO).

**Query Syntax**  SYSTem:ERRor:NEXT?

**Query Respond**  SYST:ERR:NEXT?

The following is one of the possible errors:
-109, Missing parameter

### 14.5 Local Mode

**Description**  Set the power supply to local mode.

**Command Syntax**  SYSTem:LOCal

**Example**  SYST:LOC
14.6 LXI State

Description
Set LXI status indicator state.

Command Syntax
LXI:IDENTify:STATe <state>
<state> = {ON | OFF | 0 | 1}

Query Syntax
LXI:IDENTify:STATe?

Example
LXI:IDENT:STAT ON

Query Respond
LXI:IDENT:STAT?

Returns: 1

14.7 Memory Clear

Description
Clear all the user memory and reboot the instrument.

Command Syntax
SYSTem:SECurity:IMMediate

Example
SYST:SEC:IMM

14.8 Remote Mode

Description
Set the power supply to remote mode.

Command Syntax
SYSTem:REMote

Example
SYST:REM

14.9 Remote Mode With Lock

Description
Set the power supply to remote mode and lock all the front-panel keys, including the Lock | Unlock keys.

Command Syntax
SYSTem:RWLock

Example
SYST:RWL

14.10 Remote State

Description
Set remote | local | remote with lock state.

Command Syntax
SYSTem:COMMunicate:RLSTate <state>
<state> = {LOCAL | REMote | RWLock }

Query Syntax
SYSTem:COMMunicate:RLSTate?

Example
SYST:COMM:RLST REM

Query Respond
SYST:COMM:RLST?

Returns: REM
14.11 Single BEEP

**Description**: The command issues a single beep immediately.

**Command Syntax**: SYSTem:BEEPer:IMMediate

**Example**: SYST:BEEP:IMM

14.12 Time

**Description**: Set the time of the power supply’s real-time clock.

**Command Syntax**: SYSTem:TIME <HH>,<MM>,<SS>

**Query Syntax**: SYSTem:TIME?

**Example**: SYST:TIME 12,22,32

**Query Respond**: SYST:TIME?

Returns: 12,22,32

14.13 Version

**Description**: Query the present version of the power supply.

**Query Syntax**: SYSTem:VERSion?

**Query Respond**: SYST:VERS?

Returns: 1999.0
15.1 Output Configuration

**Description** Configure the output’s: voltage, current, and protection settings.

**Example**

1. Select channel to be configured.
   - `INST:SEL 0` or `INST:SEL 1` or `INST:SEL 2`

   **Note:**
   Only one output can be configured at a time.

2. Set over current protection state (OCP). Over voltage protection cannot be disabled.
   - `SOUR:CURR:PROT:STAT {1 | 0 | ON | OFF}`

3. Set OVP/OCP limit.
   - `SOUR:VOLT:PROT:LEV:AMPL 35.2`
   - `SOUR:CURR:PROT:LEV:AMPL 8.8`

4. Set the voltage min/max.
   - `VOLT:MAX 60.6`
   - `VOLT:MIN 0`

5. Set the slew time.
   - `VOLT:SLOP 3000`
   - `CURR:SLOP 250`

6. Set the timer state.
   - `OUTP:TIM ON`

7. Set the timer setting.
   - `OUTP:TIM:COUN 1,0,0`

8. Set the voltage/current.
   - `SOUR:VOLT:LEV:IMM:AMPL 5`
   - `SOUR:CURR:LEV:IMM:AMPL 5`

9. Enable/disable output of the selected channel.
   - `OUTP:STAT {0 | 1 | OFF | ON}`
**15.2 List Edit**

**Description** Editing a list requires the commands to be sent in a sequence. If the commands are not sent in the correct sequence the unit will beep indicating it does not recognize the command.

**Additional Information** A list can be broken down into 3 parts: **List Setup**, **List Parameters**, and **Step Parameters**. List setup is independent of list parameter and step parameters.

**List Setup**

- List setup includes the following parameters:
  - **List State**
  - **List Number**
  - **List Pace**
  - **Trigger Source**
  - **After List**

  These parameters are independent and can be set at any point.

**List and Step Parameters** The list and step parameters are codependent and must be sent in a sequence for the unit to recognize the commands.

**Note:**

While editing a list’s parameters new changes will be stored in temporary memory. SOURC:LIST:SAVE must be used to save the changes to the internal memory.

**ABORT** all list initiated in any channels before you begin the sequence.

**Sequence**

1. Enter edit mode of a specified list. Before entering edit mode abort all running list. Only one list can be edited at a time.
   - SOUR:LIST:NUMB <list>; list := {1 to 10}

2. Set the list parameters:
   - SOUR:LIST:COUN <repeat>
     repeat := {1 to 99999}
   - SOUR:LIST:NEXT <next>
     next := {0|1 to 10} 0 := off

3. To edit a step enter step edit mode using:
   - SOURce:LIST:STEP:NUMBER <number>; number := {1 to 100}

4. Edit the step’s parameters (ranges may vary depending on model and output mode)
   - SOURce:LIST:VOLTage:LEVel <voltage>
     voltage := {0 to 180 V}
   - SOURce:LIST:CURRent:LEVel <current>
     current := {0 to 24 A}
   - SOURce:LIST:TOUTput:BOSTep:DATA <state>
     state := {1|0|On|Off}
   - SOURce:LIST:TOUTput:EOSTep:DATA <state>
     state := {1|0|On|Off}
   - SOURce:LIST:DWELl <time>
     time := {0|0.1 to 9999} 0 := terminate step

5. Save any changes made using the command:
   - SOURce:LIST:SAVE
Examples

To only edit the List Parameters:


To edit both the List and Step Parameters:

SOUR:LIST:NUMB 2;SOUR:LIST:COUN 1;SOUR:LIST:NEXT 5:
SOUR:LIST:STEP:NUMB 1
SOUR:LIST:CURR:LEV 1;SOUR:LIST:VOLT:LEV 1;SOURce:LIST:DWELI

Note:

When editing the step parameters all 5 parameters must be set even if no change is being made to one of them. If not all parameters are configured the unit will not accept a command to edit another step until all parameters have been set.

15.3 Running a List

Description

After setting a list’s parameters, run the list by following one of the sequence below. Example 1 demonstrated how to run a list with pace set to Trigger. Example 2 demonstrates how to run a list with pace set to Dwell.

Examples 1

<table>
<thead>
<tr>
<th>INST 0</th>
<th>select a channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTP:MODE LIST,1</td>
<td>enable and assign list</td>
</tr>
<tr>
<td>OUTP 1</td>
<td>turn output on</td>
</tr>
<tr>
<td>INIT 1</td>
<td>start list on selected channel</td>
</tr>
<tr>
<td>*TRG</td>
<td>send a trigger</td>
</tr>
<tr>
<td>INIT 1</td>
<td>abort list on selected channel</td>
</tr>
<tr>
<td>*TRG</td>
<td>exit list mode</td>
</tr>
<tr>
<td>ABOR 0</td>
<td></td>
</tr>
<tr>
<td>OUTP:MODE FIX</td>
<td>Dwell</td>
</tr>
</tbody>
</table>

Immediate Triggers

Continuous Triggers

Examples 2
15.4 Datalogger

**Description**  Set the parameters before starting the datalogger.

**Commands**

1. Set the sampling interval.
   - SENS:DLOG:FUNC:TINT 1

2. Set the trigger source.
   - TRIG:DLOG:SOUR BUS

3. Initiate the logging session.
   - INITiate:IMMediate:DLOG

4. Start the datalogger.
   - TRIGger:DLOG:IMMediate

5. Stop the datalogger.
   - ABORt:DLOG

**Example**

15.5 Operation Modes

Description
Set the operation mode to either Series, Parallel, or Tracking.

Example

1. Select operation mode { OFF | PARA2 | PARA3 | SERI2 | SERI3 | TRAC2 | TRAC3}
   - OUTP:PAIR SERI2

2. Select channel to configure the output of chosen mode.

   Note:
   If chosen mode uses All CH any one of the channel can be chosen to edit the output settings. If CH1+2 is chosen then either channel one or two must be chosen to edit the output settings.
   - INST:SEL 1  Chosen channel can vary depending on mode selected.

   Note:
   The configuration of the output is the same as the in Normal Mode.

3. Set over current protection state (OCP). Over voltage protection cannot be disabled.
   - SOUR:CURR:PROT:STAT {1 | 0 | ON | OFF}

4. Set OVP/OCP limit.

5. Set the voltage min/max .
   - VOLT:MAX 60.6; VOLT:MIN 0

6. Set the slew time .
   - VOLT:SLOP 3000; CURR:SLOP 250

7. Set the timer state.
   - OUTP:TIM ON

8. Set the timer setting.
   - OUTP:TIM:COUN 1,0,0

9. Set the voltage/current.

10. Enable/disable output of the selected channel.
   - OUTP:STAT {0 | 1 | OFF | ON}