

# Programming Manual

## 9240 Series Multi-Range DC Power Supplies



# Contents

1	About Commands & Queries	5
1.1	How They are Listed	5
1.2	How They are Described	5
1.3	When can they be used?	5
1.4	Command Notation	5
2	Common Command Introduction	6
2.1	*CLS	7
2.2	*ESE	7
2.3	*ESR?	9
2.4	*IDN?	9
2.5	*OPC	9
2.6	*PSC	10
2.7	*RCL	10
2.8	*SAV	10
2.9	*RST	11
2.10	*SRE	11
2.11	*STB?	11
2.12	*TRG	12
2.13	*TST?	12
2.14	*WAI	12
3	Voltage Subsystem	13
3.1	:MEASure[:SCALar]:VOLTage[:DC]?	13
3.2	[:SOURce]:VOLTage:PROTection[:LEVel][:AMPLitude]	13
3.3	[:SOURce]:VOLTage:MAXimum	13
3.4	[:SOURce]:VOLTage:MINimum	14
3.5	[:SOURce]:VOLTage:PROTection:CLEar	14
3.6	[:SOURce]:VOLTage:PROTection:TRIPped?	14
3.7	[:SOURce]:VOLTage:SENSe	14
3.8	[:SOURce]:VOLTage:SLOPe	14
3.9	[:SOURce]:VOLTage[:LEVel][:IMMediate][:AMPLitude]	15
4	Current Subsystem	16
4.1	:MEASure[:SCALar]:CURRent[:DC]?	16
4.2	[:SOURce]:CURRent[:LEVel][:IMMediate][:AMPLitude]	16
4.3	[:SOURce]:CURRent:SLOPe	17
4.4	[:SOURce]:CURRent:PROTection:CLEar	17
4.5	[:SOURce]:CURRent:PROTection[:LEVel][:AMPLitude]	17
4.6	[:SOURce]:CURRent:PROTection:STATe	17
4.7	[:SOURce]:CURRent:PROTection:TRIPped?	17
5	MEASure Subsystem	18
5.1	:MEASure[:SCALar]:POWer[:DC]?	18
5.2	:MEASure[:SCALar]:CURRent[:DC]?	18
5.3	:MEASure[:SCALar]:VOLTage[:DC]?	18
6	Output Subsystem	19
6.1	:OUTPut:PON:STATe	19
6.2	:OUTPut:PROTection:CLEar	19
6.3	:OUTPut[:STATe]	20
6.4	:OUTPut:TIMer	20
6.5	:OUTPut:TIMer:COUNT	20

7	Digital I/O Subsystem	21
7.1	:OUTPut:INHibit:MODE	21
7.2	[:SOURce]:DIGital:INPut:DATA?	22
7.3	[:SOURce]:DIGital:OUTPut:DATA	22
7.4	[:SOURce]:DIGital:PIN<NR1>:FUNction	23
7.5	[:SOURce]:DIGital:PIN<NR1>:POLarity	23
8	List Subsystem	24
8.1	[:SOURce]:LIST:CLEar	25
8.2	[:SOURce]:LIST:COUNt	25
8.3	[:SOURce]:LIST:CURRent[:LEVel]	25
8.4	[:SOURce]:LIST:DELeTe	26
8.5	[:SOURce]:LIST:DWELI	26
8.6	[:SOURce]:LIST:NEXT	27
8.7	[:SOURce]:LIST:NUMBer	27
8.8	[:SOURce]:LIST:SAVE	28
8.9	[:SOURce]:LIST:STATe	28
8.10	[:SOURce]:LIST:STEP	28
8.11	[:SOURce]:LIST:STEP:NUMBer	29
8.12	[:SOURce]:LIST:TERMinate:LAST	29
8.13	[:SOURce]:LIST:TOUTput:BOStep[:DATA]	30
8.14	[:SOURce]:LIST:TOUTput:EOStep[:DATA]	30
8.15	[:SOURce]:LIST:VOLTag[:LEVel]	31
8.16	Edit List Sequence	31
8.16.1	Additional Information	31
8.17	Run a List	34
9	Battery Subsystem	35
9.1	BATTery:MODEl:ENABLE	35
9.2	BATTery:MODEl:STATe	35
9.3	BATTery:MODEl:VOLTag	35
9.4	BATTery:MODEl:CURRent	36
9.5	BATTery:MODEl:END:CURRent	36
9.6	BATTery:MODEl:END:DELay	36
9.7	BATTery:CONFigure:SAMPLing	36
9.8	BATTery:CONFigure:AH	37
9.9	BATTery:CONFigure:WH	37
9.10	BATTery:FAILSAFe:STATe	37
9.11	BATTery:FAILSAFe:TIme	37
9.12	BATTery:FAILSAFe:Ah	38
9.13	BATTery:FAILSAFe:Wh	38
9.14	BATTery:FAILSAFe:OVP	38
9.15	BATTery:STATe?	39
9.16	BATTery:CONFigure:TRIGger:SOURce	39
10	Initiate Subsystem	40
10.1	:INIT:[IMMediate]	40
10.2	:INIT:[IMMediate]:DLOG	40
10.3	:INIT:CONTInuous	41
11	Trigger Subsystem	42
11.1	:TRIGger:DLOG[:IMMediate]	42
11.2	:TRIGger:DLOG:SOURce	42
11.3	:TRIGger[:SEQuence]:SOURce	43
12	Abort Subsystem	44
12.1	:ABORt	44
12.2	:ABORt:DLOG	44

13	Display Subsystem	45
13.1	:DISPlay[:WINDow][:STATe]	45
14	Instrument Subsystem	46
14.1	Apply	46
15	SENS Subsystem	47
15.1	SENSe:DLOG:FUNcTion:PERiod	47
15.2	SENSe:DLOG:FUNcTion:TINTerval	47
16	Status Subsystem	48
16.1	STATus:PRESet	48
16.2	STATus:QUEStionable:INSTrument:ENABLE	48
16.3	STATus:QUEStionable:INSTrument[:EVENT?]	48
16.4	STATus:QUEStionable:ISUMmary<NR1>[:EVENT?]	49
16.5	STATus:QUEStionable:ISUMmary:CONDition?	49
16.6	STATus:QUEStionable:ISUMmary<NR1>:ENABLE	49
16.7	STATus:QUEStionable[:EVENT]	49
16.8	STATus:QUEStionable:ENABLE	50
16.9	Status Diagrams	50
17	System Subsystem	52
17.1	:SYSTem:BEEPer[:IMMediate]	52
17.2	:SYSTem:BEEPer:STATe	52
17.3	:SYSTem:DATE	53
17.4	:SYSTem:ERRor[:NEXT]	53
17.5	:SYSTem:LOCal	53
17.6	:LXI:IDENtify	53
17.7	:SYSTem:SECurity:IMMediate	53
17.8	:SYSTem:REMote	54
17.9	:SYSTem:RWLock	54
17.10	:SYSTemRLState	55
17.11	:SYSTem:TIME	55
17.12	:SYSTem:VERSion?	55
18	Appendix	56
18.1	Output Configuration	56
18.2	List Edit	57
18.3	Running a List	58
18.4	Datalogger	59
18.5	Operation Modes	60

# 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.

---

## 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 9240 Series 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.

# Common Command Introduction

IEEE standard defines the common commands used for querying the basic inSyntax 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.

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

**Table 2.1** Common Commands

## 2.1 \*CLS

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

<b>SESR</b>	<b>(IEEE 488.2)</b>
<b>OPERation Status Register</b>	<b>(SCPI)</b>
<b>QUESTionable Status Register</b>	<b>(SCPI)</b>
<b>Error/Event Queue</b>	<b>(SCPI)</b>

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 (see 2.5, ??, and ??) 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

## 2.2 \*ESE

**Description** **Event Status Enable Command and Query.** Enables bits in the enable register for the Standard Event Register group. The selected bits are then reported to bit 5 of the Status Byte Register. The following types of events are reported: power-on detected, command syntax errors, command execution errors, self-test or calibration errors, query errors, or the \*OPC command has been executed. Any or all of these conditions can be reported to the Standard Event summary bit through the enable register. To set the enable register mask, you must write a decimal value to the register using the \*ESE command. See [Table 2.2](#)

The \*ESE? query returns a decimal value which corresponds to the binary-weighted sum of all bits enabled by the \*ESE command.

The **Standard Event Enable Register** is cleared when:

- The \*ESE 0 command is executed.
- The instrument was powered on and configured so the function generator clears the enable register using the \*PSC 1 command.
- \*CLS does not clear the enable register but it does clear all bits in the event register.
- A STATus:PRESet does not clear the bits in the Status Byte enable register.

**Note:**

The enable register will not be cleared at power-on if the function generator was configured using the **\*PSC 0** command.

**Syntax** \*ESE <enable value>

<enable value> := 0 to 128.

For example, to enable bit 2 (value 4), bit 3 (value 8), and bit 7 (value 128), the decimal sum would be 140 (4 + 8 + 128). The default decimal sum is 0.

**Query** \*ESE?

**Response** \*ESE <value>

**Example** \*ESE?

Bit	Bit Name	Decimal Value	Definition
0	OPC	1	All commands prior to and including *OPC have completed and the overlapped command (e.g., *TRG for burst) has completed.
1	not used	not used	Not used. Returns 0.
2	QYE	4	The instrument tried to read the output buffer but it was empty. Or, a new command line was received before a previous query has been read. Or, both the input and output buffers are full.
3	DDE	8	A self-test, cal, or other device-specific error has occurred.
4	EXE	16	An execution error has occurred.
5	CME	32	A command syntax error has occurred.
6	not used	64	Not used. Returns 0.
7	PON	128	Power has been cycled on since the last time the event register was read or cleared.

**Table 2.2** Standard Event Register

---

## 2.3 \*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** \*ESR <value>  
<value> := 0 to 255

**Query** \*ESR?

**Example** \*ESR?

Return: 0

**Related** \*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** \*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&KPrecision,9241,614D21108,0.30\_0825A-0.18\_0824A

---

## 2.5 \*OPC

---

**Description** Sets the **Operation Complete** bit (bit 0) in the **Standard Event Register** after all of the previous commands have been completed. Other commands may be executed before the bit is set.

This command is used to stop the controller until all pending commands are completed. **\*OPC?** returns **"1"** to the output buffer after the previous commands have been completed.

Other commands cannot be executed until this command completes.

**Syntax** \*OPC

**Query** \*OPC?

**Example** INITiate;\*OPC

**Response** "1"

---

## 2.6 \*PSC

---

**Description** **Power-On Status Clear.** Clears the Standard Event Enable Register and Status Byte Condition Register at power on (\*PSC 1). When \*PSC 0 is in effect, these two registers are not cleared.

The **\*PSC?** query returns the power-on status clear setting. Returns "0" (do not clear at power on) or "1" (clear at power on).

**Syntax** \*PSC <state>  
<state> := {0 |1}  
**Default** := \*PSC 1

**Query** \*PSC?

**Example** \*PSC 1

**Response** "1"

---

## 2.7 \*RCL

---

**Description** Recalls the instrument state stored in the specified non-volatile storage location. The instrument's state cannot be recalled from an empty storage location. When shipped from the factory, storage locations "1" through "9" are empty (location "0" has the power-on state).

After attempting to recall an empty location the message "**Failed to recall settings**" will be displayed in the top left corner of the instruments's display.

Setting OUTPut:PN:STATe to USER will load the settings saved in the selected location when the instrument is powered on.

**Syntax** \*RCL <memory address>  
<memory address> := {0 to 9}

**Example** \*RCL 1

**Related** \*SAV

---

## 2.8 \*SAV

---

**Description** Store (save) the current instrument state in the specified non-volatile storage location. Any state previously stored in the same location will be overwritten. The instrument state can be stored in any of the 10 storage locations (0-9).

An instrument reset (**\*RST** command) does not affect the configurations stored in memory. Once a state is stored, it remains until it is overwritten or specifically deleted.

If OUTPut:PN:STATe is set to LAST the settings set when the instrument is powered off will be savein the selected User Settings, overwriting any previous saved settings in this location.

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

**Example** \*SAV 0

**Related** \*RCL

---

## 2.9 \*RST

---

**Description** Reset the instrument to its factory default state. \*RST does not affect stored instrument states, or the I/O settings, which are stored in non-volatile memory.

**Syntax** \*RST

**Example** \*RST

---

## 2.10 \*SRE

---

**Description** The Status Byte Summary Register reports conditions from the other status registers. Data waiting in the instrument's output buffer is immediately reported on the "**Message Available**" bit (bit 4). Clearing an event register from one of the other register groups will clear the corresponding bits in the **Status Byte Condition Register**. Reading all messages from the output buffer, including any pending queries, will clear the "**Message Available**" bit.

To enable specific bits, you must write a decimal value which corresponds to the binary-weighted sum of the bits in the register. The selected bits are summarized in the "**Master Summary**" bit (bit 6) of the Status Byte Register. If any of the selected bits change from "**0**" to "**1**", a Service Request Signal (SRQ) is generated.

The **\*SRE?** query returns a decimal value which corresponds to the binary-weighted sum of all bits enabled by the **\*SRE** command

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 its returned value is always zero.

- **\*CLS** (clear status) does not clear the enable register but it does clear all bits in the event register.
- **STATus:PRESet** does not clear the bits in the Status Byte enable register.
- **\*PSC 0** preserves the contents of the enable register through power cycles.

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

**Query** \*SRE?

**Example** \*SRE 4

**Response** "4"

---

## 2.11 \*STB?

---

**Description** Query the summary (condition) register in this register group. This command is similar to a Serial Poll but it is processed like any other instrument command. This command returns the same result as a Serial Poll but the "**Master Summary**" bit (bit 6) is not cleared by the **\*STB?** command.

**Query** \*STB?

**Example** \*STB?

**Response** 0 to 255

**Related** \*CLS, \*SRE

Bit	Bit Name	Decimal Value	Definition
0 to 2	not used	not used	Not used. Returns 0.
3	QUES	8	One or more bits are set in the <b>Questionable Status Register</b> . (bits must be enabled)
4	MAV	16	Data is available in the instrument's output buffer.
5	ESB	32	One or more bits are set in the <b>Standard Event Register</b> . (bits must be enabled)
6	RQS	64	The Service Request Line (SRQ) on the GPIB is designed to signal the Controller when a service request is pending.
7	OPER	128	One or more bits are set in the <b>Operation Status Byte Register</b> . (bits must be enabled)

Table 2.3 Status Byte Register

## 2.12 \*TRG

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

**Syntax** \*TRG

**Example** \*TRG

**Related** TRIGger:SOURce

## 2.13 \*TST?

**Description** The \*TST? query performs an internal self-test of the power supply. Returns "0" (PASS) or "1" (FAIL). If the test fails, one or more error messages will be generated to provide additional information on the failure. Use the SYSTem:ERRor? command to read the error queue

**Query** \*TST?

**Example** \*TST?

**Response** "0"

## 2.14 \*WAI

**Description** Wait for all pending operations to complete before executing any additional commands over the interface.

**Syntax** \*WAI

**Example** The following command string guarantees that the first trigger is accepted and the operation is executed before the second trigger is recognized.

**TRIG:SOUR BUS;\*TRG;\*WAI;\*TRG;\*WAI**

**Related** \*OPC?

# Voltage Subsystem

The VOLTage subsystem controls the amplitude characteristics of the source.

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

**[:SOURce]:VOLTage:PROTection[:LEVel][:AMPLitude]**

**[:SOURce]:VOLTage:MAXimum**

**[:SOURce]:VOLTage:MINimum**

**[:SOURce]:VOLTage:PROTection:CLEAr**

**[:SOURce]:VOLTage:PROTection:TRIPped?**

**[:SOURce]:VOLTage:SENSe**

**[:SOURce]:VOLTage:SLOPe**

**[:SOURce]:VOLTage[:LEVel][:IMMEDIATE][:AMPLitude]**

---

## 3.1 :MEASure[:SCALar]:VOLTage[:DC]?

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

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

**Response** MEAS:SCAL:VOLTage:DC?

Returns: The output's voltage.

---

## 3.2 [:SOURce]:VOLTage:PROTection[:LEVel][:AMPLitude]

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

**Syntax** [:SOURce]:VOLTage:PROTection[:LEVel][:AMPLitude] <NRf>

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

**Example** VOLT:PROT 30

**Response** <NRf>

---

## 3.3 [:SOURce]:VOLTage:MAXimum

**Description** Set maximum voltage output.

**Syntax** [:SOURce]:VOLTage:MAXimum <NRf>

**Query** VOLTage:MAXimum?

**Example** VOLT:MAX 60.6

**Response** <NRf>

---

### 3.4 [:SOURce]:VOLTage:MINimum

---

**Description** Set minimum voltage output.

**Syntax** [:SOURce]:VOLTage:MINimum <NRf>

**Query** VOLTage:MINimum?

**Example** VOLT:MIN 0

**Response** <NRf>

---

### 3.5 [:SOURce]:VOLTage:PROTection:CLEar

---

**Description** Clears the tripped OVP (Over Voltage Protection).

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

**Example** VOLT:PROT:CLE

---

### 3.6 [:SOURce]:VOLTage:PROTection:TRIPped?

---

**Description** Query the state of OVP.

**Query** [:SOURce]:VOLTage:PROTection:TRIPped?

**Response** <bool>

"0" if OVP is not tripped

"1" if OVP is tripped

---

### 3.7 [:SOURce]:VOLTage:SENSe

---

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

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

**Query** [:SOURce]:VOLTage:SENS?

**Example** VOLT:SENS ON

**Response** <bool>

"0" if voltage sense is enabled

"1" if voltage sense is disabled

---

### 3.8 [:SOURce]:VOLTage:SLOPe

---

**Description** Set voltage slew.

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

**Query** [:SOURce]:VOLTage:SLOPe?

**Example** VOLT:SLOP 3000

**Response** NRf

---

### 3.9 [:SOURce]:VOLTage[:LEVel][:IMMEDIATE][:AMPLitude]

---

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

**Syntax** [:SOURce]:VOLTage[:LEVel][:IMMEDIATE][:AMPLitude] <NRf>

**Query** [:SOURce]:VOLTage[:LEVel][:IMMEDIATE][:AMPLitude?]

**Example** VOLT 5

**Response** <NRf>

# Current Subsystem

The CURRent subsystem controls the amplitude characteristics of the source.

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

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

**[:SOURce]:CURRent:SLOPe**

**[:SOURce]:CURRent:PROTection:CLEAr**

**[:SOURce]:CURRent:PROTection[:LEVel][:AMPLitude]**

**[:SOURce]:CURRent:PROTection:STATe**

**[:SOURce]:CURRent:PROTection:TRIPped?**

---

## 4.1 :MEASure[:SCALar]:CURRent[:DC]?

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

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

**Response** MEAS:SCAL:CURRent:DC?

Returns: The output's current.

---

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

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

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

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

**Example** CURR 1

**Response** <NRf>

---

### 4.3 [[:SOURce]:CURRent:SLOPe

---

**Description** Set current slew.

**Syntax** [[:SOURce]:CURRent:SLOPe <NRf>

**Query** [[:SOURce]:CURRent:SLOPe?

**Example** CURR:SLOP 3000

**Response** <NRf>

---

### 4.4 [[:SOURce]:CURRent:PROTection:CLEar

---

**Description** Clears the tripped OCP (Over Current Protection).

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

**Example** CURR:PROT:CLE

---

### 4.5 [[:SOURce]:CURRent:PROTection[:LEVel][:AMPLitude]

---

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

**Syntax** [[:SOURce]:CURRent:PROTection[:LEVel][:AMPLitude] <NRf>

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

**Example** CURR:PROT 4

**Response** <NRf>

---

### 4.6 [[:SOURce]:CURRent:PROTection:STATe

---

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

**Syntax** [[:SOURce]:CURRent:PROTection:STATe <bool>  
<bool> := { 0 or OFF := disable, 1 or ON := enable}

**Query** [[:SOURce]:CURRent:PROTection:STATe?

**Example** CURR:PROT:STAT OFF

**Response** <bool>

---

### 4.7 [[:SOURce]:CURRent:PROTection:TRIPped?

---

**Description** Query the state of OCP.

**Query** [[:SOURce]:CURRent:PROTection:TRIPped?

**Response** <bool>

**0** if OCP is not tripped

**"1"** if OCP is tripped

# MEASure Subsystem

The following commands query the measured value of voltage, current, power, or all three measurements combined.

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

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

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

---

## 5.1 :MEASure[:SCALar]:POWer[:DC]?

**Description** Query measured power.

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

**Example** MEAS:POW?

**Response** <NRf>

---

## 5.2 :MEASure[:SCALar]:CURRent[:DC]?

**Description** Query the current voltage.

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

**Response** <NRf>

---

## 5.3 :MEASure[:SCALar]:VOLTage[:DC]?

**Description** Query the measured voltage.

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

**Response** <NRf>

# Output Subsystem

The following commands set and query the output's parameters as well as the power-on parameters.

**:OUTPut:PON:STATe**

**:OUTPut:PROTection:CLEar**

**:OUTPut[:STATe]**

**:OUTPut:TIMer**

**:OUTPut:TIMer:COUNT**

---

## 6.1 :OUTPut:PON:STATe

---

**Description** Sets the power on state. Specifies which file to load when the instrument is powered on. **OFF** will load the default parameters. **LAST** will load the parameters that were set before the instrument was last powered off. **USER** will load the specified saved file stored in the internal memory. The file location must be specified when **USER** mode is selected.

**Syntax** OUTPut:PON:STATe <string>,<NR1>

<bool> := {OFF, LAST, USER,@ programmed user>}

<NR1> := {0 to 9}

**Query** OUTPut:PON:STATe?

**Example** User mode memory location 2 :=: OUT:PON:STAT USER,2

LAST mode := OUTP:PON STAT LAST

**Response** <string> when **OFF** or **LAST** mode is selected.

<string>,<NR1> when **USER** mode is selected.

---

## 6.2 :OUTPut:PROTection:CLEar

---

**Description** Clear all triggered protections.

**Syntax** OUTPut:PROTection:CLEar

**Example** OUTP:PROT:CLE

---

### 6.3 :OUTPut[:STATe]

---

**Description** Enable/disable the output.

**Syntax** OUTPut[:STATe] <bool>  
<bool> := {0 |1 |OFF |ON}

**Query** OUTPut:[STATe?]

**Example** OUTP:STAT 1

**Response** <bool>

---

### 6.4 :OUTPut:TIMer

---

**Description** Enable/disable the output timer.

**Syntax** OUTPut:TIMer <bool>  
<bool> := {0 |1 |OFF |ON}

**Query** OUTPut:TIMer?

**Example** OUTP:TIM 1

**Response** <bool>

**Related** OUTPut:TIMer:COUNT

---

### 6.5 :OUTPut:TIMer:COUNT

---

**Description** Set the dwell time of the output. **Timer State** must be enabled.

**Syntax** OUTPut:TIMer:COUNT <NR1>,<NR2>,<NR3>  
<NR1> := {0 to 99}; hours  
<NR2> := {0 to 59}; minutes  
<NR3> := {0 to 59}; seconds

**Query** OUTPut:TIMer:COUNT?

**Example** OUTP:TIM:COUNT 1,30,00

**Response** <NR1>,<NR2>,<NR3>

# Digital I/O Subsystem

The following commands set and/or query the **Digital I/O** parameters.

**:OUTPut:INHibit:MODE**

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

**[[:SOURce]:DIGital:OUTPut:DATA**

**[[:SOURce]:DIGital:PIN<NR1>:FUNCtion**

**[[:SOURce]:DIGital:PIN<NR1>:POLarity**

---

## 7.1 :OUTPut:INHibit:MODE

---

**Description** Sets the inhibit input function when pin 3 is configured as a remote inhibit input. To configure pin 3 as a remote inhibit use the command **SOUR:DIG:PIN3:FUNC INH**.

**Syntax** OUTPut:INHibit:MODE <string>  
<string> := {LATChing |LIVE |OFF}

**Query** OUTPut:INHibit:MODE?

**Example** OUTP:INH:MODE LIVE

**Response** <string>

**Related** [[:SOURce]:DIGital:PIN<pin number>:FUNCtion

### Note:

Inhibit mode must be enabled. See section [7.4](#)

---

## 7.2 [:SOURce]:DIGital:INPut:DATA?

**Description** Query the input of all 3 Digital\_In pins. For Digital Input function, a voltage high input with positive polarity will return a bit 1 at the pin while a voltage low input with negative polarity will return a bit 0 at the pin. A voltage high input with negative polarity will return a bit 0 at the pin while a voltage low input with negative polarity will return a bit 1 at the pin.

Pin 1 is the least significant bit, while pin 3 is the most significant bit.

### Note:

If the pins function is not set to **Digital\_IN** when the query is sent, the pin will return the last input before the function was switched out of **Digital\_In**. Therefore, the polarity and input will not affect the binary bit returned.

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

**Example** DIG:INP:DATA?

If the voltage input in all pins is high and the polarity for all pins is positive the instrument will return "7".  
 If the voltage input in all pins is high and the polarity for all pins is negative the instrument will return "0".  
 If the voltage input in pins 1 and 3 is high and the voltage at pin 2 is low with positive polarity for all pins the instrument will return "5"

```
0 := 000
1 := 001
2 := 010
3 := 011
4 := 100
5 := 101
6 := 110
7 := 111
```

**Response** <NR1>

## 7.3 [:SOURce]:DIGital:OUTPut:DATA

**Description** For Digital Output function, a binary bit 1 with positive polarity specify a voltage high at the pin while a binary bit 0 with positive polarity specify a voltage low at the pin. Data is programmed as followed:  
 Pin 1 is the least significant bit, while pin 3 is the most significant bit.

**Syntax** [:SOURce]:DIGital:OUTPut:DATA <NR1> <NR1> := {0 to 7}

**Query** [:SOURce]:DIGital:OUTPut:DATA?

**Example** DIG:OUTP:DATA 7

To send a binary weighted value configure pins 1 through 3 as 111 (7) if all polarities are set to pos.

**Response** <NR1>

---

## 7.4 [:SOURce]:DIGital:PIN<NR1>:FUNcTion

---

**Description** Set the function of the selected pins.

**Syntax** [:SOURce]:DIGital:PIN<NR1>:FUNcTion <string>

<NR1> := { 1, 2, or 3 }

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

### Note:

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

---

**Query** [:SOURce]:DIGital:PIN<NR1>:FUNcTion?

**Example** DIG:PIN1:FUNC NONE

**Response** <string>

---

## 7.5 [:SOURce]:DIGital:PIN<NR1>:POLarity

---

**Description** Set the polarity of the selected pin.

**Syntax** [:SOURce]:DIGital:PIN<NR1>:POLarity <string>

<NR1> := { 1 | 2 }

<string> := { POS | NEG }

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

**Example** DIG:PIN1:POL POS

**Response** <string>

# List Subsystem

The list subsystem controls automatic sequencing through the following SCPI commands:

**[[:SOURce]:LIST:CLEar**

**[[:SOURce]:LIST:COUNT**

**[[:SOURce]:LIST:CURRent[:LEVel]**

**[[:SOURce]:LIST:DELeTe**

**[[:SOURce]:LIST:DWELI**

**[[:SOURce]:LIST:NEXT**

**[[:SOURce]:LIST:NUMBer**

**[[:SOURce]:LIST:SAVE**

**[[:SOURce]:LIST:STATe**

**[[:SOURce]:LIST:STEP**

**[[:SOURce]:LIST:STEP:NUMBer**

**[[:SOURce]:LIST:TERMinate:LAST**

**[[:SOURce]:LIST:TOUTput:BOStep[:DATA]**

**[[:SOURce]:LIST:TOUTput:EOStep:DATA**

**[[:SOURce]:LIST:VOLTage[:LEVel]**

**Run a List**

**Edit List Sequence**

## 8.1 [:SOURce]:LIST:CLEAr

**Description** Clears all the steps and List Run Parameters of the selected list.

**Syntax** [:SOURce]:LIST:CLEAr

**Example** :LIST:NUMB 1::LIST:CLE::LIST:SAVE

All parameters of the selected list (1) 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

**Related** [:SOURce]:LIST:NUMBER  
[:SOURce]:LIST:SAVE

### Note:

Before clearing all parameters enter **List Edit Mode** by sending the command [:SOURce]:LIST:NUMBER <NR1>. Where <NR1> is the list to be configured. After clearing the list the changes must be saved using the command [:SOURce]:LIST:SAVE.

## 8.2 [:SOURce]:LIST:COUNt

**Description** Sets the list repeat count. This sets the number of times that a list is executed before it completes. Default is 0.

**Syntax** [:SOURce]:LIST:COUNt <NR1>

<NR1> = {1 to 100000}

**Query** [:SOURce]:LIST:COUNt?

**Example** LIST:NUMB 1::LIST:COUN 10::LIST:SAVE

**Response** <NR1>

**Related** [:SOURce]:LIST:NUMBER  
[:SOURce]:LIST:SAVE

## 8.3 [:SOURce]:LIST:CURRent[:LEVel]

**Description** Set the current level for the selected list step.

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

<NRf> := {0 to 24 A}

### Note:

Current output range will vary depending on instrument model and selected operation mode.

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

**Example** LIST:CURR 2

**Response** LIST:CURR?

Returns: <NRf>

**Related** [:SOURce]:LIST:STEP:NUMBer

[:SOURce]:LIST:NUMBer

---

## 8.4 [:SOURce]:LIST:DELeTe

---

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

**Syntax** [:SOURce]:LIST:DELeTe <NR1>

<NR1> = {1 to 100}

**Example** :LIST:NUMB 1;:LIST:DEL 10;: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.

**Related** [:SOURce]:LIST:NUMBer  
[:SOURce]:LIST:SAVE

### Note:

Before deleting steps enter **List Edit Mode** by sending the command [:SOURce]:LIST:NUMBer <NR1>. Where <NR1> is the list to be configured. After sending the delete command the changes must be saved using the command [:SOURce]:LIST:SAVE.

---

---

## 8.5 [:SOURce]:LIST:DWELL

---

**Description** Set the dwell time for the selected list step.

**Syntax** [:SOURce]:LIST:DWELL <NR2>

<NRf> := {0.1 to 9999999.0 s}

**Query** [:SOURce]:LIST:DWELL?

**Example** LIST:DWELL 2.0

**Response** LIST:DWELL?

Returns: <NR2>

**Related** [:SOURce]:LIST:STEP:NUMBer

[:SOURce]:LIST:NUMBer

---

## 8.6 [:SOURce]:LIST:NEXT

---

**Description** Set the list to be executed when the current list elapses. A list can run infinitely by setting next to the list that is currently running.

**Syntax** [:SOURce]:LIST:NEXT <NR1>

<NR1> := {0 to 10}

**0** := no list will run after current list elapses

**1 to 10** := List that will run after current list elapses.

**Query** [:SOURce]:LIST:NEXT?

**Example** :LIST:NUMB 1;:LIST:NEXT 0;:LIST:SAVE  
No list will run once list 1 completes.

**Response** <NR1>

**Related** [:SOURce]:LIST:NUMBER  
[:SOURce]:LIST:SAVE

### Note:

Before setting **Next** enter **List Edit Mode** by sending the command [:SOURce]:LIST:NUMBER <NR1>. Where <NR1> is the list to be configured. After sending the [:SOURce]:LIST:NEXT <NR1> command the changes must be saved using the command [:SOURce]:LIST:SAVE.

---

---

## 8.7 [:SOURce]:LIST:NUMBER

---

**Description** Enable the **Edit List** mode of the selected list. When edit list mode is enabled the list count and list next parameters can be configured. The instrument can enter **Edit Step** mode once **Edit List** mode has been enabled.

### Note:

Changes made in **Edit List** mode must be saved using the command [:SOURce]:LIST:SAVE.

---

**Syntax** [:SOURce]:LIST:NUMBER <NR1>

<NR1> := {1 to 10}

**Query** [:SOURce]:LIST:NUMBER?

**Example** SOUR:LIST:NUMB 2

**Response** SOUR:LIST:NUMB?

Returns: <NR1>

---

## 8.8 [:SOURce]:LIST:SAVE

---

**Description** Save data set in temporary memory to the selected list.  
**To configure any of the list parameters all running list must first be aborted.**

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

**Example** LIST:SAVE

**Related** [:SOURce]:LIST:STEP:NUMBer

[:SOURce]:LIST NUMBER

ABORt

---

## 8.9 [:SOURce]:LIST:STATe

---

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

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

**Example** LIST:STAT?

**Response** LIST:STAT?  
Returns: <bool>

**Related** :SOURce]:LIST:STEP:NUMBer

---

## 8.10 [:SOURce]:LIST:STEP

---

**Description** Set the pace of the 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. The trigger signal required will vary based on the selected list trigger source.

**Syntax** [:SOURce]:LIST:STEP <bool>

<bool> := {0 | 1 | OFF | ON}

0 | OFF := Dwell

1 | ON := Trigger

**Query** [:SOURce]:LIST:STEP?

**Example** SOUR:LIST:STEP ON

**Response** SOUR:LIST:STEP?  
Returns: <bool>

**Related** :TRIGger[:SEQuence]:SOURce

---

## 8.11 [:SOURce]:LIST:STEP:NUMBer

---

**Description** Enables the **Edit Step** mode of the selected step. When **Edit Step** mode is enabled the steps parameters: **Voltage, Current, BOST, EOST and Dwell** can be configured. To enter **Edit Step** the instrument must first be in **Edit List** mode.

### Note:

Changes made in **Edit Step** mode must be saved using the command **[:SOURce]:LIST:SAVE**.

---

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

<NR1> := {1 to 100}

**Query** [:SOURce]:LIST:STEP:NUMBer?

**Example** :LIST:STEP:NUMB 2

**Response** :LIST:STEP:NUMB?

Returns: <NR1>

**Related** **[:SOURce]:LIST:SAVE**

**[:SOURce]:LIST:NUMBer**

---

## 8.12 [:SOURce]:LIST:TERMinate:LAST

---

**Description** Set the list terminate state. When terminate state is set to **Last** the output's parameters will remain at the last list value. When the terminate state is set to **DC** the output's parameters will returns to the DC value that was in effect before the output sequence started.

**Syntax** [:SOURce]:LIST:TERMinate:LAST <bool>

<bool> := {0 | OFF | 1 | ON} 0 | OFF := DC

1 | ON := Last

**Query** [:SOURce]:LIST:TERMinate:LAST?

**Example** LIST:TERM:LAST ON

**Response** <bool>

### 8.13 [:SOURce]:LIST:TOUTput:BOSTep[:DATA]

**Description** Enable/disable a trigger-out signal at the beginning of the selected step (BOST).

**Syntax** [:SOURce]:LIST:TOUTput:BOSTep[:DATA] <bool>

<voltage> := {0 |1}

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

**Example** LIST:TOUT:BOST 1

**Response** <bool>

#### Note:

In order to enable/disable BOST a step must be selected first. If a step is not selected the instrument will beep notifying that there was an issue programming the BOST. For more information see Editing/Creating a List

### 8.14 [:SOURce]:LIST:TOUTput:EOSTep[:DATA]

**Description** Enable/disable a trigger-out signal at the end of the selected step (EOST).

**Syntax** [:SOURce]:LIST:TOUTput:EOSTep[:DATA] <bool>

<bool> := {0 |1}

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

**Example** LIST:TOUT:EOST 1

**Response** <bool>

#### Note:

In order to enable/disable EOST a step must be selected first. If a step is not selected the instrument will beep notifying that there was an issue programming the EOST. For more information see Editing/Creating a List

---

## 8.15 [:SOURce]:LIST:VOLTage[:LEVel]

---

**Description** Set the voltage level for the selected list step.

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

<NRf> := {0 to 24 V}

### Note:

Voltage output range will vary depending on instrument model and selected operation mode.

---

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

**Example** LIST:VOLT 2

**Response** LIST:VOLT?

Returns: <NRf>

**Related** [:SOURce]:LIST:STEP:NUMBer

[:SOURce]:LIST:STEP:NUMBer

---

## 8.16 Edit List Sequence

---

Editing a list requires the commands to be sent in a sequence. Certain commands require the instrument to be in either **Edit List** or **Edit Step** mode. If such command is sent when the instrument is not in the require mode the instrument will beep indicating it does not recognize the command.

### 8.16.1 Additional Information

---

A list can be broken down into 3 parts: **List Setup**, **List Parameters**, and **Step Parameters**:

#### List Setup

---

List setup is independent of List Parameters and Step Paramters. It includes the following parameters:

List State   **List Number**   **Pace**   **Trigger Source**   **After List**

These parameters are independent and can be configured at any point in any mode.

## List Parameters

To configure the list parameters the instrument must enter **Edit List** mode. To enter edit list mode:

- Abort all running list.
  - Use the **:ABORT** command to abort the list running on the selected channel. `:INStrument[:SElect]`.
- Send the command `[:SOURce]:LIST:NUMBer` to enter **Edit List** mode.

In **Edit List** mode the following parameters can be configured:

**Next Repeat**

### Note:

Parameters configured in **Edit List** mode are saved in temporary memory. Exiting the edit list mode before saving will cause all changes to be lost. To permanently save the changes use the command `[:SOURce]:LIST:SAVE`.

## Step Parameters

To configure the **Step Parameters** the instrument must first enter **Edit Step** mode. **Edit Step** mode can only be entered from the **Edit List** mode. Please refer to section [sec:LIST Parameters] for instructions on how to enter **Edit List** mode.

Once the instrument is in **Edit List** mode send the command `[:SOURce]:LIST:STEP:NUMBer` to enter **Edit Step** mode. Upon entering **Edit Step** mode all parameters must be configured before being able to save the changes or proceeding to a different step. If all parameters are not configured the changes will not be saved. If a parameter's value will remain the same sending the command with the same value will allow the all changing to be saved.

In **Edit Step** mode the following parameters can be configured:

**Voltage Current BOST EOST Dwell**

### Note:

Parameters configured in **Edit List** mode are saved in temporary memory. Exiting the edit list mode before saving will cause all changes to be lost. To permanently save the changes use the command `[:SOURce]:LIST:SAVE`.

## 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 <NR1>                      ▪ SOUR:LIST:NEXT <next>
  - <NR1> := {1 to 99999}                      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>                      ▪ SOURce:LIST:CURRent:LEVel <current>
  - voltage:= {0 to 180 V}                      current := {0 to 24 A}
  - SOURce:LIST:TOUTput:BOSTep:DATA <state>                      ▪ SOURce:LIST:TOUTput:EOSTep:DATA <state>
  - state := {1|0|On|Off}                      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**:

```
SOUR:LIST:NUMB 2;;SOUR:LIST:COUN 4;;SOUR:LIST:NEXT 4;;SOUR:LIST:SAVE
```

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:DWELl
```

```
SOUR:LIST:TOUT:BOST:DATA 1;;SOUR:LIST:TOUT:EOST:DATA 0;;SOUR:LIST:SAVE
```

### 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.

## 8.17 Run 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

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

**Immediate Triggers**

INST 0	<b>select a channel</b>
OUTP:MODE LIST,1	<b>enable and assign list</b>
OUTP 1	<b>turn output on</b>
INIT 0	<b>start list on selected channel</b>
INIT:CONT 0,1	<b>enable continuous triggering</b>
*TRG	<b>send a trigger</b>
*TRG	
*TRG	
*TRG	
ABOR 0	<b>abort list on selected channel</b>
OUTP:MODE FIX	<b>exit list mode</b>

**Continuous Triggers**

### Examples 2

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

**Dwell**

# Battery Subsystem

The following commands set or query the parameters related to the battery test system.

## Note:

Mode series parameters can only be used when enabled. To enable **Battery Charge Mode** use the command **BATTery:MODEl:ENABLE**.

---

## 9.1 BATTery:MODEl:ENABLE

---

**Description** Enter or exit battery charge mode. When enabled, the screen switches to the **Battery Charge Mode** page.

**Syntax** BATTery:MODEl:ENABLE <bool>

<bool> := {0 | 1 or ON | OFF}

**Query** BATTery:MODEl:ENABLE?

**Example** BATT:MOD:ENAB 1

**Response** <bool>

## 9.2 BATTery:MODEl:STATE

---

**Description** Start or stop the battery test.

**Syntax** BATTery:MODEl:STATE <bool>

<bool> := {0 | 1 or ON | OFF}

**Query** BATTery:MODEl:STATE?

**Example** BATT:MOD:STAT 1

**Response** <bool>

## 9.3 BATTery:MODEl:VOLTage

---

**Description** Sets the V full value.

**Syntax** BATTery:MODEl:VOLTage <NR2>

<NR2> := {0.000 to max voltage output}

**Query** BATTery:MODEl:VOLTage?

**Example** BATT:MOD:VOLT 14.500

**Response** <NR2>

---

## 9.4 BATTery:MODEl:CURRent

---

**Description** Sets the I full value. Limits the current value that can be sourced.

**Syntax** BATTery:MODEl:CURRent <NR2>  
<NR2> := {0.000 to max current rating}

**Query** BATTery:MODEl:CURRent?

**Example** BATT:MOD:CURR 2.700

**Response** <NR2>

---

## 9.5 BATTery:MODEl:END:CURRent

---

**Description** Sets the I end value. The charging test will end once the I End value is reached.

**Syntax** BATTery:MODEl:END:CURRent <NR2>  
<NR2> := {0.000 to max current rating}

**Query** BATTery:MODEl:END:CURRent?

**Example** BATT:MOD:END:CURR 0.090

**Response** <NR2>

---

## 9.6 BATTery:MODEl:END:DELay

---

**Description** Sets a delay to end the charging test once the I end value is reached.

**Syntax** BATTery:MODEl:END:DELay <NR1>  
<NR1> := {0 to 255 seconds}

**Query** BATTery:MODEl:END:DELay

**Example** BATT:MOD:END:DEL 10

**Response** <NR1>

---

## 9.7 BATTery:CONFigure:SAMPLing

---

**Description** Sets the sampling time(sec) for the charge test logging.

**Syntax** BATTery:CONFigure:SAMPLing <NR2>  
<NR2> := {0.5 to 300.0 seconds}

**Query** BATTery:CONFigure:SAMPLing?

**Example** BATT:CONF:SAMPL 1.0

**Response** <NR2>

---

## 9.8 BATTery:CONFigure:AH

---

**Description** Enable/disable Ah logging.

**Syntax** BATTery:CONFigure:AH <bool>

<bool> := {0 | 1 or ON | OFF}

**Query** BATTery:CONFigure:AH?

**Example** BATT:CONF:AH 1

**Response** <bool>

---

## 9.9 BATTery:CONFigure:WH

---

**Description** Enable/disable Wh logging.

**Syntax** BATTery:CONFigure:WH <bool>

<bool> := {0 | 1 or ON | OFF}

**Query** BATTery:CONFigure:WH?

**Example** BATT:CONF:WH 1

**Response** <bool>

---

## 9.10 BATTery:FAILSAFe:STATe

---

**Description** Enable/disable the the fail safe protection.

**Syntax** BATTery:FAILSAFe:STATe <bool>

<bool> := {0 | 1 or ON | OFF}

**Query** BATTery:FAILSAFe:STATe?

**Example** BATT:FAILSAF:STAT 1

**Response** <bool>

---

## 9.11 BATTery:FAILSAFe:TIME

---

**Description** Set the fail safe time.

**Syntax** BATTery:FAILSAFe:TIME <HH>,<MM>,<SS>

<HH> := { 0 to 99}

<MM> := { 0 to 99 }

<SS> := { 0 to 99 }

**Query** BATTery:FAILSAFe:TiMe?

**Example** BATT:FAILSAF:TIM 01,00,00

**Response** <NR1>,<NR1>,<NR1>

---

## 9.12 BATTery:FAILSAFe:Ah

---

**Description** Sets the Ah fail save value.

**Syntax** BATTery:FAILSAFe:Ah <NR2>

<NR2> := { 0.00 to 9999.99 }

**Query** BATTery:FAILSAFe:Ah?

**Example** BATT:FAILSAF:Ah 100.00

**Response** <NR2>

---

## 9.13 BATTery:FAILSAFe:Wh

---

**Description** Sets the Wh fail save value.

**Syntax** BATTery:FAILSAFe:Wh <NR2>

<NR2> := { 0.00 to 9999.99 }

**Query** BATTery:FAILSAFe:Wh?

**Example** BATT:FAILSAF:Wh 100.00

**Response** <NR2>

---

## 9.14 BATTery:FAILSAFe:OVP

---

**Description** Sets the OVP fail save value.

**Syntax** BATTery:FAILSAFe:OVP

**Query** BATTery:FAILSAFe:OVP?

**Example** BATT:FAILSAF:OVP 14.7

**Response** <NR2>

---

## 9.15 BATTery:STATe?

---

**Description** Checks the status of the battery test. If the test is in progress it returns 1. If the test is idle it returns 0.

**Query** BATTery:STATe?

**Example** BATT:STATe?

**Response** <bool>

---

## 9.16 BATTery:CONFigure:TRIGger:SOURce

---

**Description** Sets the battery test trigger source

**Syntax** BATTery:CONFigure:TRIGger:SOURce <NR1 or string>

<NR1 or string> := { 0 or IMMEDIATE | 1 or EXTERNAL | 2 or BUS }

**Query** BATTery:CONFigure:TRIGger:SOURce?

**Example** BATTery:CONFi:TRIG:SOUR 1

**Response** <string>

# Initiate Subsystem

The **INITiate Subsystem** is used to control the initiation of the trigger subsystem. It initiates the trigger sequence of the selected function. The 914X series contains two sequences that can be initiated with this subsystem, List Mode and Data Logger.

**:INIT:[IMMediate]**

**:INIT:[IMMediate]:DLOG**

**:INIT:CONTInuous**

---

## 10.1 :INIT:[IMMediate]

**Description** Initiates the trigger sequence for list mode when list mode is enabled. Only the trigger sequence of the selected channel will be enabled.

**Syntax** INITiate[:IMMediate] <NR1>

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

0 := channel 1

1 := channel 2

2 := channel 3

**Example** INIT 0

**Related** :OUTPut:MODE

---

## 10.2 :INIT:[IMMediate]:DLOG

**Description** Initiates the trigger sequence for the data logger. If trigger source is set to manual INIT:DLOG will initiate the action and trigger the event. If the data logging trigger source is set to External(I/O) or BUS(Remote) the action will only be initiated the trigger signal is must be sent to begin recording.

A USB drive must be connected to the instrument's USB host port. If no USB drive is connected the instrument will beep indicating the command was not read, and error code -200 will be returned.

-200,Unknown error

**Syntax** INITiate[:IMMediate]:DLOG

**Example** INIT:DLOG

**Related** :TRIGger:DLOG:SOURce

:TRIGger:DLOG[:IMMediate]

\*TRG

---

### 10.3 :INIT:CONTInuous

---

**Description** Sets the trigger system to either continuously initiate or initiate only once. With **CONTInuous** disabled the trigger system will remain in the IDLE state until **CONTInuous** is enabled or INITiate:IMMEDIATE is received.

With **CONTInuous** enabled the trigger system will exit IDLE state, and on completion of each trigger cycle, the trigger system will immediately commence another trigger cycle without entering the IDLE state.

**Syntax** INITiate:CONTInuous <channel>,<bool>

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

0 := channel 1

1 := channel 2

2 := channel 3

<state> := {0 |1}

**Example** INIT:CONT 0,1

# Trigger Subsystem

The trigger subsystem is used to synchronize device actions with events. A device action might be the acquisition of a measurement of the application of a stimulus. To perform a device action with a trigger signal the action must first be armed. Arming an action will remove the device from IDLE state and into the wait for trigger state. The arming of an action is affected by either the **INITiate[:IMMEDIATE]** command or by setting **INITiate:CONTinuous** to ON.

The following commands can be used to configure the trigger source of an action or to trigger said action:

**:TRIGger:DLOG[:IMMEDIATE]**

**:TRIGger:DLOG:SOURce**

**:TRIGger[:SEQUence]:SOURce**

---

## 11.1 :TRIGger:DLOG[:IMMEDIATE]

---

**Description** Generates an immediate data logger trigger when the trigger source is set to BUS.

**Syntax** TRIGger:DLOG[:IMMEDIATE]

**Example** TRIG:DLOG

**Related** **:INIT[:IMMEDIATE]:DLOG**  
**TRIGger:DLOG:SOURce**

---

## 11.2 :TRIGger:DLOG:SOURce

---

**Description** Configures the trigger source for the **Data Logger**. When Manual control is selected the instrument will wait for a trigger signal sent using the corresponding softkey. When remote control is selected the instrument will wait for bus trigger (\*TRG). When digital I/O control is selected the instrument will wait for a trigger signal received in the corresponding digital I/O pin.

**Syntax** TRIGger:DLOG:SOURce <string>  
<string> := {IMMEDIATE|BUS|EXTERNAL}

**IMM** := Manual control

**BUS** := Remote control

**EXT** := Digital I/O control

**Query** TRIGger:DLOG:SOURce?

**Example** TRIG:DLOG:SOUR BUS

**Response** TRIG:DLOG:SOUR?

Returns: <string>

**Related** TRIGger:DLOG[:IMMEDIATE]  
:INIT[:IMMEDIATE]:DLOG

---

### 11.3 :TRIGger[:SEQuence]:SOURce

---

**Description** Configures the trigger source for **List** mode. When Manual control is selected the instrument will wait for a trigger signal sent using the corresponding softkey. When remote control is selected the instrument will wait for bus trigger (\*TRG). When digital I/O control is selected the instrument will wait for a trigger signal received in the corresponding digital I/O pin.

**Syntax** TRIGger[:SEQuence]:SOURce <string>  
  
<string> := {IMMEDIATE|BUS|EXTernal}

**IMM** := Manual control

**BUS** := Remote control

**EXT** := Digital I/O control

**Query** TRIGger[:SEQuence]:SOURce?

**Example** TRIG:SOUR BUS

**Response** TRIG:SOUR?

Returns: <string>

**Related** :INIT[:IMMEDIATE]  
:INIT:CONTinuous

# Abort Subsystem

The ABORT subsystem resets the trigger system and places all trigger sequences in the IDLE state. Any actions related to the trigger system that are in progress will be aborted as quickly as possible. The ABORT commands are not considered complete until all trigger sequences are in the IDLE state. The execution of an ABORT command sets the pending operation flags that were set by the initiation of the trigger system to false.

## Note:

This command is an event and has no associated \*RST condition or query form.

**:ABORt**

**:ABORt:DLOG**

## 12.1 :ABORt

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

**Syntax** ABORt <NR1>

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

0 := Channel 1

1 := Channel 2

2 := Channel 3

**Example** ABOR 0

## 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.

## 12.2 :ABORt:DLOG

**Description** If recording is in progress the command will end data logging and return the trigger system to IDLE. If data logging is initiated but waiting for a trigger the command returns the trigger system to IDLE.

**Syntax** ABORt:DLOG

**Example** ABOR:DLOG

# Display Subsystem

the DISPLAY subsystem controls the presentation of of textual and graphical 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.

**:DISPlay[:WINDow][:STATe]**

---

## 13.1 :DISPlay[:WINDow][:STATe]

---

**Description** Disable/enable the display update. Disabling the display will cause the display to remain static. The keys will not be lock, meaning that inputs will still be register but the display will not be updated to reflect the input.

**Syntax** DISPlay:WINDow:STATe <bool>

<bool> := {1 | ON | 0 | OFF}

**Query** DISPlay:WINDow:STATe?

**Example** DISP:WIND:STAT ON

**Response** DISP:WIND:STAT?

Returns: <bool>

# Instrument Subsystem

## 14.1 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

# SENS Subsystem

The SENSE subsystem contains commands to configure the sample period and time interval of the data acquisition.

**SENSe:DLOG:FUNction:PERiod**

**SENSe:DLOG:FUNction:TINTerval**

---

## 15.1 SENSe:DLOG:FUNction:PERiod

---

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

**Syntax** SENSe:DLOG:FUNction:PERiod <NR2>

<NR2> := {.2 to 300 seconds}

**Query** SENSe:DLOG:FUNction:PERiod ?

**Example** SENS:DLOG:FUNC:PER 1.0

**Response** SENS:DLOG:FUNC:PER?

Returns: <NR2>

---

## 15.2 SENSe:DLOG:FUNction:TINTerval

---

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

**Syntax** SENSe:DLOG:FUNction:TINTerval <NR2>

<NR2> := {.2 to 300 seconds}

**Query** SENSe:DLOG:FUNction:TINTerval ?

**Example** SENS:DLOG:FUNC:TINT 1.0

**Response** SENS:DLOG:FUNC:TINT?

Returns: <NR2>

# Status Subsystem

The STATus subsystem controls the SCPI-defined status-reporting structures ( QUESTIONable, OPERation, Instrument SUMmary and INSTRument registers). These registers are comprised of condition registers, an event register, and an 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.

---

## 16.1 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** STATus:PRESet

**Example** STAT:PRESet

---

## 16.2 STATus:QUESTIONable:INSTRument:ENABLE

---

**Description** Sets the enable mask which allows true conditions in the event register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the associated summary bit.

**Syntax** STATus:QUESTIONable:INSTRument:ENABLE <NR1>

**Query Format** STATus:QUESTIONable:INSTRument:ENABLE?

**Example** STAT:QUES:INST:ENAB 1

**Response** STAT:QUES:INST:ENAB?

Returns: <NR1>

---

## 16.3 STATus:QUESTIONable:INSTRument[:EVENT?]

---

**Description** This query returns the contents of the event register associated with the status structure defined in the command.

**Query** STATus:QUESTIONable:INSTRument[:EVENT?]

**Response** STAT:QUES:INST?

Returns: <NR1>

**Note:**

Reading the event register clears it.

## 16.4 STATus:QUESTionable:ISUMmary<NR1>[:EVENT?]

**Description** This query returns the contents of the selected channel's event register associated with the status structure defined in the command.

**Query** STATus:QUESTionable:ISUMmary<NR1>[:EVENT?]

**Response** STAT:QUES:ISUM2?

Returns: <NR1>

See table Questionable Status Summary in [16.9](#)

## 16.5 STATus:QUESTionable:ISUMmary:CONDition?

**Description** This query returns the CV or CC condition of the specified channel. Reading the condition register is non-destructive. The response is NR1: 1 indicating CC mode and 2 indicating CV mode.

**Query** STATus:QUESTionable:ISUMmary:CONDition?

**Response** STAT:QUES:ISUM:COND?

Returns: <NR1>

## 16.6 STATus:QUESTionable:ISUMmary<NR1>:ENABLE

**Description** Sets the enable mask which allows true conditions in the selected channel's event register to be reported in the summary bit. If a bit is 1 in the enable register and its associated event bit transitions to true, a positive transition will occur in the associated summary bit.

**Syntax** STATus:QUESTionable:ISUMmary<NR1>:ENABLE <NR1>

**Query Format** STATus:QUESTionable:ISUMmary<NR1>:ENABLE?

**Example** STAT:QUES:ISUM:ENAB 1

**Response** STAT:QUES:ISUM:ENAB?

Returns: <NR1>

## 16.7 STATus:QUESTionable[:EVENT]

**Description** This query returns the contents of the event register associated with the status structure defined in the command.

**Note:**

Reading the event register clears it.

**Query** STATus:QUEStionable[:EVENT]?

**Example** STAT:QUES:EVEN?  
Returns: <NR1>

## 16.8 STATus:QUEStionable:ENABLE

**Description** Set questionable status enable register.

**Syntax** STATus:QUEStionable:ENABLE <NR1>

**Query Format** STATus:QUEStionable:ENABLE?

**Example** STATus:QUEStionable:ENABLE 1

**Response** STAT:QUES:ENAB?  
Returns: <NR1>

## 16.9 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 + .

Bit	Bit Name	Decimal Value	Definition
0	VOLTage	1	The power supply is operating in constant voltage mode.
1	CURRent	2	The power supply is operating in constant current mode.
2	OVP	4	OVP (Over Voltage Protection) has been triggered.
3	OCP	8	OCP (Over Current Protection) has been triggered.
4	OTP	16	OTP (Over Temperature Protection) has been triggered.
5	UNR	32	The output is unregulated.
6	RNC	64	Remote sense is enabled and not connected.
7	WTG_DLOG	128	DLOG is waiting for a trigger.
8	WTG	256	The instrument is waiting for a trigger.
9	INH	512	Inhibit has occurs.

**Table 16.1** Questionable Status Summary

Bit	Bit Name	Decimal Value	Definition
0	OPC	1	All commands prior to and including *OPC have completed and the overlapped command (e.g., *TRG for burst) has completed.
1	not used	not used	Not used. Returns 0.
2	QYE	4	The instrument tried to read the output buffer but it was empty. Or, a new command line was received before a previous query has been read. Or, both the input and output buffers are full.
3	DDE	8	A self-test, cal, or other device-specific error has occurred.
4	EXE	16	An execution error has occurred.
5	CME	32	A command syntax error has occurred.
6	not used	64	Not used. Returns 0.
7	PON	128	Power has been cycled on since the last time the event register was read or cleared.

Table 16.2 Standard Event Register

Bit	Bit Name	Decimal Value	Definition
0 to 2	not used	not used	not used
3	QUES	8	One or more bits are set in the Questionable Data Register. Bits must be enabled, see STATUS:QUESTionable:ENABLE.
4	MAV	16	Data is available in the instrument's output buffer.
5	ESB	32	One or more bits are set in the status byte register and may generate a service request.
6	RQS	64	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.
7	OPER	128	One or more bits are set in the operation status register.

Table 16.3 Status Byte

# System Subsystem

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

**:SYSTem:BEEPer[:IMMEDIATE]**

**:SYSTem:BEEPer:STATe**

**:SYSTemRLSTATe**

**:SYSTem:DATE**

**:SYSTem:ERRor[:NEXT]**

**:SYSTem:LOCal**

**:SYSTem:REMOte**

**:SYSTem:RWLock**

**:SYSTem:SECurity:IMMEDIATE**

**:SYSTem:TIME**

**:SYSTem:VERSion?**

**:LXI:IDENTify**

---

## 17.1 :SYSTem:BEEPer[:IMMEDIATE]

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

**Syntax** SYSTem:BEEPer[:IMMEDIATE]

**Example** SYST:BEEP

---

## 17.2 :SYSTem:BEEPer:STATe

**Description** This command controls the audible beeper of the instrument.

**Syntax** SYSTem:BEEPer:STATe <bool>  
<bool> = {ON |OFF |1 |0}

**Query** SYSTem:BEEPer:STATe?

**Example** SYST:BEEP:STAT 0

**Response** SYST:BEEP:STAT?

Returns: <bool>

---

### 17.3 :SYSTem:DATE

---

**Description** Programs the date of the power supply's real-time clock. \*RST does not affect the programmed value. All parameters are entered in NR1 format.

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

**Query** SYSTem:DATE?

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

**Response** SYST:DATE?

Returns: <NR1>,<NR1>,<NR1>

---

### 17.4 :SYSTem:ERRor[:NEXT]

---

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

**Query** SYSTem:ERRor[:NEXT]?

**Response** SYST:ERR?

<NR1>,<string>

---

### 17.5 :SYSTem:LOCal

---

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

**Syntax** SYSTem:LOCal

**Example** SYST:LOC

---

### 17.6 :LXI:IDENtify

---

**Description** Sets LXI status indicator state.

**Syntax** LXI:IDENtify:STATe <bool>  
<bool> = {ON |OFF |0 |1}

**Query** LXI:IDENtify[:STATe]?

**Example** LXI:IDEN ON

**Response** LXI:IDEN?

Returns: <bool>

---

### 17.7 :SYSTem:SECurity:IMMediate

---

**Description** Clear all the user memory and reboot the instrument.

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

Table 17.1 Error Codes

**Syntax** SYSTem:SECurity:IMMEDIATE

**Example** SYST:SEC:IMM

## 17.8 :SYSTem:REMOte

**Description** Set the power supply in remote mode.

**Syntax** SYSTem:REMOte

**Example** SYST:REM

## 17.9 :SYSTem:RWLOCK

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

**Syntax** SYSTem:RWLOCK

**Example** SYST:RWL

---

## 17.10 :SYSTemRLSTate

---

**Description** Set remote |local |remote with lock state.

**Syntax** SYSTem:COMMunicate:RLSTate <string>  
<string> = {LOCAl |REMote |RWLock }

**Query** SYSTem:COMMunicate:RLSTate?

**Example** SYST:COMM:RLST REM

**Response** SYST:COMM:RLST?

Returns: <string>

---

## 17.11 :SYSTem:TIME

---

**Description** Programs the time of the power supply's real-time clock. \*RST does not affect the programmed value. All parameters are entered in NR1 format.

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

**Query** SYSTem:TIME?

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

**Response** SYST:TIME?

Returns: <NR1>,<NR1>,<NR1>

---

## 17.12 :SYSTem:VERSion?

---

**Description** This query returns the SCPI version number for which the instrument complies.

**Query** SYSTem:VERSion?

**Response** SYST:VERS?

Returns: <NR2>

# Appendix

## 18.1 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 configure 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}

## 18.2 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

- 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}
- Set the list parameters:
  - SOUR:LIST:COUN <repeat>      repeat := {1 to 99999}
  - SOUR:LIST:NEXT <next>      next := { 0|1 to 10} 0 := off
- To edit a step enter step edit mode using:
  - SOURce:LIST:STEP:NUMBER <number>; number := {1 to 100}
- 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
- Save any changes made using the command:

- SOURce:LIST:SAVE

## Examples

---

To only edit the **List Parameters**:

```
SOUR:LIST:NUMB 2;;SOUR:LIST:COUN 4;;SOUR:LIST:NEXT 4;;SOUR:LIST:SAVE
```

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
```

```
SOUR:LIST:TOUT:BOST:DATA 1;;SOUR:LIST:TOUT:EOST:DATA 0;;SOUR:LIST:SAVE
```

### 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.

---

## 18.3 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

INST 0	select a channel
OUTP:MODE LIST,1	enable and assign list
OUTP 1	turn output on
INIT 1	start list on selected channel
*TRG	send a trigger
INIT 1	
*TRG	
INIT 1	
*TRG	
ABOR 0	abort list on selected channel
OUTP:MODE FIX	exit list mode

**Immediate Triggers**

INST 0	select a channel
OUTP:MODE LIST,1	enable and assign list
OUTP 1	turn output on
INIT 0	start list on selected channel
INIT:CONT 0,1	enable continuous triggering
*TRG	send a trigger
*TRG	
*TRG	
*TRG	
ABOR 0	abort list on selected channel
OUTP:MODE FIX	exit list mode

**Continuous Triggers**

**Examples 2**

INST 0	select a channel
OUTP:MODE LIST,1	enable and assign list
OUTP 1	turn output on
INIT 0	start list on selected channel
ABOR 0	abort list on selected channel
OUTP:MODE FIX	exit list mode

**Dwell**

**18.4 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**

---

SENS:DLOG:FUNC:TINT 1;:TRIG:DLOG:SOUR BUS;:INITiate:IMMediate:DLOG;:TRIGger:DLOG:IMMediate

---

## **18.5 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.
    - SOUR:VOLT:PROT:LEV:AMPL60;;SOUR:CURR:PROT:LEV:AMPL 5
  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.
    - SOUR:VOLT:LEV:IMM:AMPL 50;;SOUR:CURR:LEV:IMM:AMPL 1
  10. Enable/disable output of the selected channel.
    - OUTP:STAT {0 |1 |OFF |ON}

**Version: January 26, 2023**