Section 5

Introduction to SCPI commands

In this section:

Introduction to SCPI

The Standard Commands for Programmable Instruments (SCPI) standard is a syntax and set of commands that is used to control test and measurement devices.

The following information describes some basic SCPI command information and how SCPI is used with the Model 2450 and presented in the Model 2450 documentation.

Command execution rules

Command execution rules are as follows:

- Commands execute in the order that they are presented in the command message.
- An invalid command generates an event message and is not executed.
- Valid commands that precede an invalid command in a command message are executed.
- Valid commands that follow an invalid command in a command message are ignored.

Command messages

A command message is made up of one or more command words sent by the controller to the instrument.

SCPI commands contain several command words that are structured to create command messages. The command words are separated by colons (`:`). For example, to configure an ethernet connection, the command words are:

```
:SYSTem:COMMunication:LAN:CONFigure
```

Many commands have query options. If there is a query option, it is created by adding a question mark (?) to the command. For example, to query the present ethernet settings, send:

```
:SYSTem:COMMunication:LAN:CONFigure?
```

Commands often take parameters. Parameters follow the command words and a space. For example, to set the instrument to automatically detect the ethernet settings, send:

```
:SYSTem:COMMunication:LAN:CONFigure AUTO
```

SCPI can also use common commands, which consist of an asterisk (*) followed by three or four letters. For example, you can reset the instrument by sending the following command:

```
*RST
```

The examples above show commands that are sent individually. You can also group command messages when you send them to the instrument. To group a set of commands, separate them with semicolons and include a colon before each command (unless it starts with an *). For example, to reset the instrument, enable relative offset for the current function, and set a relative offset of 0.5 for the current function, send the command:

```
*RST; :SENSe:CURRent:REL:STAT ON; :SENSe:CURRent:RELative .5
```

If commands are not combined, the colon (:) at the beginning of a command is optional. For example, the following commands are equivalent:

```
:SENSe:CURRent:REL:STAT ON
SENSe:CURRent:REL:STAT ON
```

If the next command in a multiple command message is on the same path, you do not need to send the colon to reset the path parsing of the command. For example, to enable relative offset and set a relative offset of 0.5 for the current function, send the command:

```
:SENSe:CURRent:RELative 0.5; REL:STAT ON
```

You can also do multiple queries in a single command message with or without resetting the path. For example, to query for the current relative offset and state, you can send:

```
:SENSe:CURRent:RELative?; :SENSe:CURRent:REL:STAT?
```

You can also send:

```
SENSe:CURRent:RELative?; rel:STAT?
```

Each new command message resets the parser path as if it was sent with the leading colon. The output for both queries is:

```
0.5;0
```

SCPI command programming notes

This section contains general information about using Standard Commands for Programmable Instruments (SCPI).

SCPI command formatting

This section describes the formatting that this manual uses when discussing SCPI commands.

SCPI command short and long forms

This documentation shows SCPI commands with both uppercase and lowercase letters. The uppercase letters are the required elements of a command. The lowercase letters are optional. However, if you choose to include the letters that are shown in lowercase letters, you must include all of them.

When you send a command to the instrument, case is not important — you can mix uppercase and lowercase letters in program messages.

For example, you can send the command `SENSe:COUNt` in any of the following formats:

```
SENSe:COUNt
sense:count
SENS:COUN
Sens:Coun
```

Optional command words

If a command word is enclosed in brackets (`[ ]`), the command word is optional. Do not include the brackets if you send the optional command word to the instrument.

For example, you can send the command `:SYSTem:BEEPer[:IMMediate] <n1>, <n2>` in any of the following formats:

```
:SYSTem:BEEPer:IMMediate 500, 1
:SYSTem:BEEPer 500, 1
:SYST:BEEP:IMMediate 500, 1
:SYST:BEEP 500, 1
```

MINimum, MAXimum, and DEFault

You can use `MINimum`, `MAXimum`, or `DEFault` instead of a parameter for some commands.

For example, you can set the parameter for the command `[:SENSe[1]]:RESistance:NPLCycles` to the minimum, maximum, or default value. To set NPLC to the minimum value, you can send either of these commands:

```
:SENSe1:RESistance:NPLCycles MINimum
:SENS:RES:NPLC MIN
```

Queries

Some commands are queries and others have a query option. These commands have a question mark (?) after the command. You can use the query to determine the present value of the parameters of the command or to get information from the instrument.

For example, to determine what the present setting for NPLC is, you can send:

```
:SENSe1:RESistance:NPLCycles?
```

This query returns the present setting.

If the command has `MINimum`, `MAXimum`, and `DEFault` options, you can use the query command to determine what the minimum, maximum, and default values are. In these queries, the ? is placed before the `MINimum`, `MAXimum`, or `DEFault` parameter. For example, to determine the default value for NPLC, you can send:

```
:SENSe1:RESistance:NPLCycles? DEFault
```

If you send two query commands without reading the response from the first, and then attempt to read the second response, you may receive some data from the first response followed by the complete second response. To avoid this, do not send a query command without reading the response. When you cannot avoid this situation, send a device clear before sending the second query command.

When you query a Boolean option, the instrument returns a 0 or 1, even if you sent OFF or ON when you originally sent the command.

SCPI parameters

The parameters of the SCPI commands are shown in angle brackets (< >). For example:

```
:SYSTem:BEEPer[:IMMediate] <frequency>, <time>
```

The type of information that you can use to replace `<frequency>` and `<time>` is defined in the Usage section of the command description. For this example, the Usage is:

`<frequency>`	The frequency of the beep (20 to 8000)
`<time>`	The amount of time to play the tone in seconds (0.001 to 100)

For this example, you can generate an audible sound by sending:

```
:SYSTem:BEEPer 500, 1
```

Note that you do not include the angle brackets when sending the command.

Sending strings

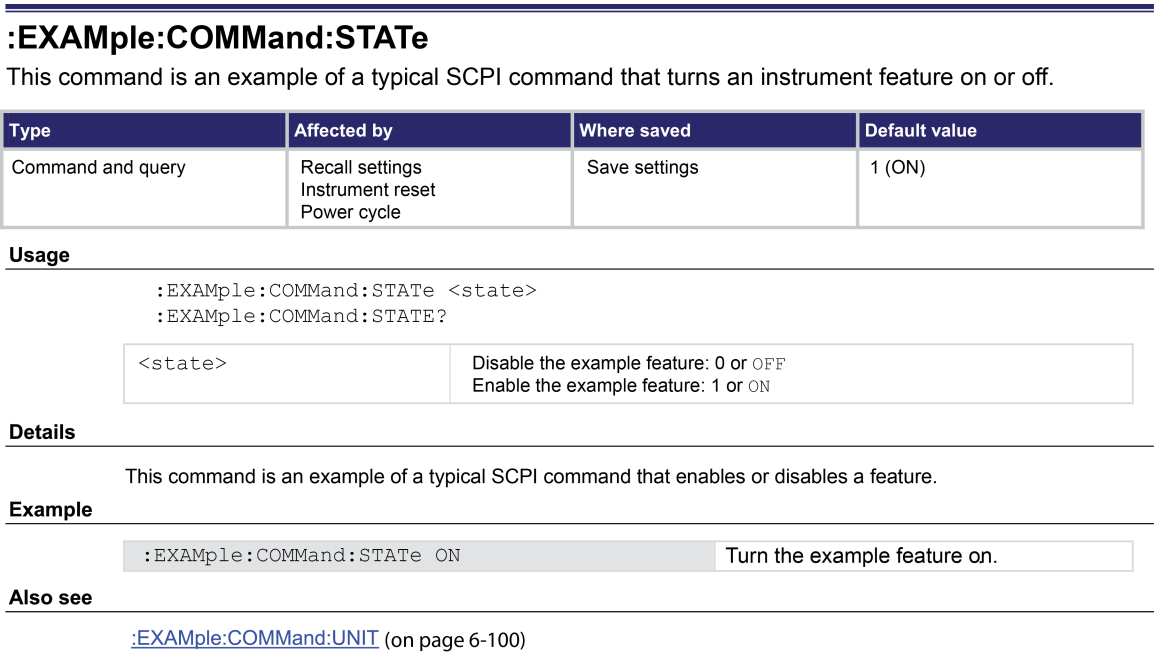
If you are sending a string, it must begin and end with matching quotes (either single quotes or double quotes). If you want to include a quote character as part of the string, type it twice with no characters in between.

A command string sent to the instrument must terminate with a <new line> character. The IEEE-488.2 EOI (end-or-identify) message is interpreted as a <new line> character and can be used to terminate a command string in place of a <new line> character. A <carriage return> followed by a <new line> is also accepted. Command string termination will always reset the current SCPI command path to the root level.

Using the SCPI command reference

The SCPI command reference contains detailed descriptions of each of the SCPI commands that you can use to control your instrument. Each command description is broken into several standard subsections. The figure below shows an example of a command description.

Figure 137: SCPI command description example



:EXAMple:COMMand:STATe

This command is an example of a typical SCPI command that turns an instrument feature on or off.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	1 (ON)

Usage

```
:EXAMple:COMMand:STATe <state>
:EXAMple:COMMand:STATE?
```

`<state>`	Disable the example feature: 0 or `OFF` Enable the example feature: 1 or `ON`

Details

This command is an example of a typical SCPI command that enables or disables a feature.

Example

`:EXAMple:COMMand:STATe ON`	Turn the example feature on.

Also see

:EXAMple:COMMand:UNIT (on page 6-100)

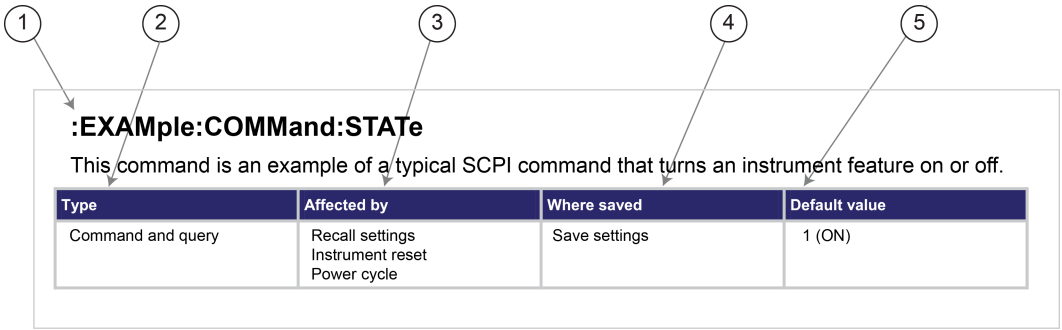
Each command listing is divided into five major subsections that contain information about the command:

- Command name and summary table
- Usage
- Details
- Example
- Also see

The content of each of these subsections is described in the following topics.

Command name and summary table

Each instrument command description starts with the command name, followed by a table with relevant information for each command. Definitions for the numbered items below are listed following the figure.



1 **Instrument command name**. Signals the beginning of the command description and is followed by a brief description of what the command does.

2 **Type of command**. Options are:

- **Command only**. There is a command but no query option for this command.
- **Command and query**. The command has both a command and query form.
- **Query only**. This command is a query.

3 **Affected by**. Commands or actions that have a direct effect on the instrument command.

- **Recall settings.** If you send `*RCL` to recall the system settings, this setting is changed to the saved value.
- **Instrument reset.** When you reset the instrument, this command is reset to its default value. Reset can be done from the front panel or when you send `*RST`.
- **Power cycle.** When you power cycle the instrument, this command is reset to its default value.
- **Source configuration list**. If you recall a source configuration list, this setting changes to the stored setting.
- **Measure configuration list.** If you recall a measure configuration list, this setting changes to the stored setting.

4 **Where saved**. Indicates where the command settings reside once they are used on an instrument. Options include:

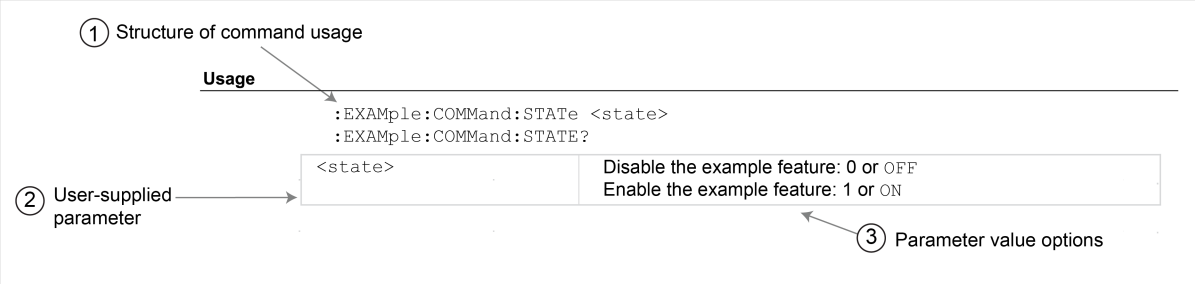
- **Not saved.** Command is not saved and must be sent each time you use it.
- **Nonvolatile memory.** The command is stored in a storage area in the instrument where information is saved even when the instrument is turned off.
- **Save settings.** This command is saved when you send the `*SAV` command.
- **Source configuration list.** This command is stored in source configuration lists.
- **Measure configuration list.** This command is stored in measure configuration lists.

5 **Default value:** Lists the default value for the command. The parameter values are defined in the Usage or Details sections of the command description.

Command usage

The Usage section of the remote command listing shows how to properly structure the command. Each line in the Usage section is a separate variation of the command usage; all possible command usage options are shown here.

Figure 138: SCPI command description usage identification



1. **Structure of command usage:** Shows how the parts of the command should be organized.
2. **User-supplied parameters:** Indicated by angle brackets (< >).

NOTE

Some commands have optional parameters. Optional parameters are presented on separate lines in the Usage section, presented in the required order with each valid permutation of optional parameters. For example:

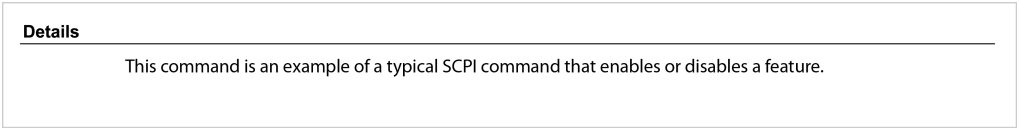
```
:SYSTem:COMMunication:LAN:CONFigure AUTO
:SYSTem:COMMunication:LAN:CONFigure MANual, IPaddress
:SYSTem:COMMunication:LAN:CONFigure MANual, IPaddress, NETmask
:SYSTem:COMMunication:LAN:CONFigure MANual, IPaddress, NETmask, GATeway
:SYSTem:COMMunication:LAN:CONFigure?
```

3. **Parameter value options:** Descriptions of the options that are available for the parameter.

Command details

This section lists additional information you need to know to successfully use the command.

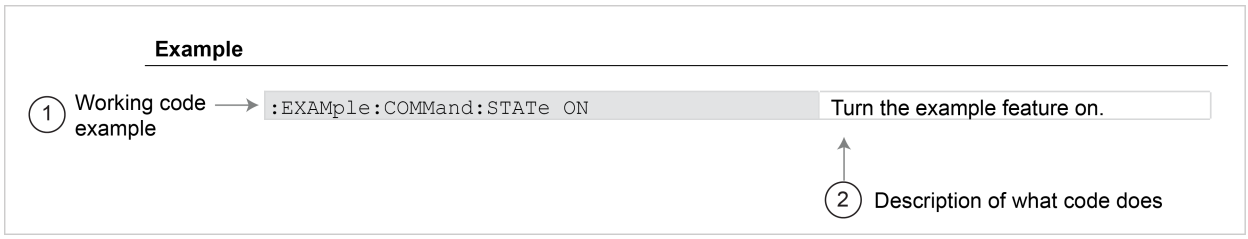
Figure 139: Details section of command listing



Example section

The Example section of the command description shows some simple examples of how the command can be used.

Figure 140: SCPI command description code examples



1. Example code that you can copy from this table and paste into your own application. Examples are generally shown using the short forms of the commands.
2. Description of the code and what it does. This may also contain the output of the code.

Related commands list

The **Also see** section of the remote command description provides links to commands that are related to the command that is being described.

Figure 141: SCPI related commands list example

Acquiring readings using SCPI commands

The following table summarizes SCPI commands that acquire and return readings.

Command	Description
FETCh?	Returns the specified data elements from the most recent reading. This command does not trigger source-measure operations. This command can repeatedly return the same readings. Until there are new readings, this command continues to return the old readings.
READ?	Makes measurements, places them in a reading buffer, and returns the specified data elements from the latest reading. The READ? query returns the same information as the following commands: TRACe:TRIGger FETCh? Do not use INITiate with the READ? command. For example, send the following command to obtain the source voltage, measured current, and relative timestamp: READ? 1, 10, "defbuffer1", SOUR, READ, REL See :READ? (on page 6-6) for more information about the READ? command.
MEASure?	Same as READ?. See :MEASure? (on page 6-4) for more information.
TRACe:DATA?	Returns specified data elements from a specified reading buffer. Use this command if SENSe:COUNT > 1. Use TRACe:TRIGger to start making measurements if you are not using the trigger model. For example, send the following command to get the source voltage, measured current, and relative timestamp: TRAC:DATA? 1, 10, "defbuffer1", SOUR, READ, REL See :TRACe:DATA? (on page 6-124) for more information about the TRACe? command.

Section 6

SCPI command reference

In this section:

:FETCh?

This query command requests the latest reading from a reading buffer.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:FETCh?
:FETCh? "<bufferName>"
:FETCh? "<bufferName>", <bufferElements>
```

`<bufferName>`	The name of the buffer where the reading is stored; if nothing is specified, `defbuffer1` is used
`<bufferElements>`	See **Details**; default is `READing`

Details

This command requests the last available reading from a reading buffer. If you send this command more than once and there are no new readings, the returned values will be the same.

NOTE

To change the number of digits returned in a remote command reading, use the `:FORMat:ASCii:PRECision` command.

You can send `:FETCh?` while a trigger model is running.

When specifying buffer elements, you can:

- Specify buffer elements in any order.
- Include any or all of the buffer elements listed below in a single list. You can repeat elements as long as the number of elements in the list is less than 14.
- Use a comma to delineate multiple elements for a data point.

The options for `<bufferElements>` are described in the following table.

Option	Description
`DATE`	The date when the data point was measured
`FORMatted`	The measured value as it appears on the front panel
`FRACtional`	The fractional seconds for the data point when the data point was measured
`READing`	The measurement reading based on the SENS:FUNC setting; if no buffer elements are defined, this option is used
`RELative`	The relative time when the data point was measured
`SEConds`	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
`SOURce`	The source value; if readback is ON, then it is the readback value, otherwise it is the programmed source value (see :SOURce[1]:<function>:READ:BACK (on page 6-87))
`SOURFORMatted`	The source value as it appears on the display
`SOURSTATus`	The status information associated with sourcing. The values returned indicate the status of the following conditions: • Overvoltage protection was active • Measured source value was read • Overtemperature condition existed • Source function level was limited • Four-wire sense was used • Output was on
`SOURUNIT`	The unit of value associated with the source value
`STATus`	The status information associated with the measurement; see the "Buffer status bits for sense measurements" table below
`TIME`	The time for the data point
`TSTamp`	The timestamp for the data point
`UNIT`	The unit of measure associated with the measurement

The output of :FETCh? is affected by the data format selected by :FORMat[:DATA]. If you set FORMat[:DATA] to REAL or SREAL, you will have fewer options for buffer elements. The only buffer elements available are READing and SOURce. If you request a buffer element that is not permitted for the selected data format, the instrument generates the error 1133, "Parameter 4, Syntax error, expected valid name parameters."

The STATus buffer element returns status values for the readings in the buffer. The status values are floating-point numbers that encode the status value. Refer to the following table for values.

Buffer status bits for sense measurements

Bit (hex)	Name	Decimal	Description
0x0001	STAT_QUESTIONABLE	1	Measure status questionable
0x0006	STAT_ORIGIN	6	A/D converter from which reading originated; for the Model 2450, this will always be 0 (Main)
0x0008	STAT_TERMINAL	8	Measure terminal, front is 1, rear is 0
0x0010	STAT_LIMIT2_LOW	16	Measure status limit 2 low
0x0020	STAT_LIMIT2_HIGH	32	Measure status limit 2 high
0x0040	STAT_LIMIT1_LOW	64	Measure status limit 1 low
0x0080	STAT_LIMIT1_HIGH	128	Measure status limit 1 high
0x0100	STAT_START_GROUP	256	First reading in a group

Example

`FETCh? "defbuffer1", DATE, READ`	Retrieve the date and measurement value for the most recent data captured in defbuffer1. Example output: 03/21/2013,-1.375422E-11

Also see

:FORMat[:DATA] (on page 6-38)
:INITiate[:IMMediate] (on page 6-146)
:MEASure? (on page 6-4)
:READ? (on page 6-6)
:TRACe:DATA? (on page 6-124)
:TRACe:TRIGger (on page 6-141)

:MEASure?

This command makes measurements, places them in a reading buffer, and returns the last reading.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:MEASure?
:MEASure:<function>?
:MEASure:<function>? "<bufferName>"
:MEASure:<function>? "<bufferName>", <bufferElements>
:MEASure? "<bufferName>"
:MEASure? "<bufferName>", <bufferElements>
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<bufferName>`	The name of the buffer where the reading is stored; if nothing is specified, `defbuffer1` is used
`<bufferElements>`	See **Details**

Details

This command makes a measurement using the specified function and stores the reading in a reading buffer.

If you do not define the function parameter, the instrument uses the presently selected measure function.

This query makes the number of readings specified by `[:SENSe[1]]:COUNt`. When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

If you define a specific reading buffer, the reading buffer must exist before you make the measurement.

To get multiple readings, use the `:TRACe:DATA?` command.

Sending this command changes the measurement function to the one specified by <function>. This function remains selected after the measurement is complete.

:MEASure? performs the same function as READ?.

:MEASure:<function>? performs the same function as sending :SENse:FUNCtion, then READ?.

NOTE

To change the number of digits returned in a remote command reading, use the :FORMat:ASCii:PRECision command.

When specifying buffer elements, you can:

- Specify buffer elements in any order.
- Include any or all of the buffer elements listed below in a single list. You can repeat elements as long as the number of elements in the list is less than 14.
- Use a comma to delineate multiple elements for a data point.

The options for <bufferElements> are described in the following table.

Option	Description
DATE	The date when the data point was measured
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds for the data point when the data point was measured
READing	The measurement reading based on the SENS:FUNC setting; if no buffer elements are defined, this option is used
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
SOURce	The source value; if readback is ON, then it is the readback value, otherwise it is the programmed source value (see :SOURce[1]:<function>:READ:BACK (on page 6-87))
SOURFORMatted	The source value as it appears on the display
SOURSTATus	The status information associated with sourcing. The values returned indicate the status of the following conditions: • Overvoltage protection was active • Measured source value was read • Overtemperature condition existed • Source function level was limited • Four-wire sense was used • Output was on
SOURUNIT	The unit of value associated with the source value
STATus	The status information associated with the measurement; see the "Buffer status bits for sense measurements" table below
TIME	The time for the data point
TSTamp	The timestamp for the data point
UNIT	The unit of measure associated with the measurement

The output of :MEASure? is affected by the data format selected by :FORMat[:DATA]. If you set FORMat[:DATA] to REAL or SREAL, you will have fewer options for buffer elements. The only buffer elements available are READing and SOURce. If you request a buffer element that is not permitted for the selected data format, the instrument generates the error 1133, "Parameter 4, Syntax error, expected valid name parameters."

The STATus buffer element returns status values for the readings in the buffer. The status values are floating-point numbers that encode the status value. Refer to the following table for values.

Buffer status bits for sense measurements

Bit (hex)	Name	Decimal	Description
0x0001	STAT_QUESTIONABLE	1	Measure status questionable
0x0006	STAT_ORIGIN	6	A/D converter from which reading originated; for the Model 2450, this will always be 0 (Main)
0x0008	STAT_TERMINAL	8	Measure terminal, front is 1, rear is 0
0x0010	STAT_LIMIT2_LOW	16	Measure status limit 2 low
0x0020	STAT_LIMIT2_HIGH	32	Measure status limit 2 high
0x0040	STAT_LIMIT1_LOW	64	Measure status limit 1 low
0x0080	STAT_LIMIT1_HIGH	128	Measure status limit 1 high
0x0100	STAT_START_GROUP	256	First reading in a group

Example

```
TRACe:MAKE "voltMeasBuffer", 10000
MEAS:VOLT? "voltMeasBuffer", FORM, DATE, READ
```

Create a buffer named voltMeasBuffer. Make a voltage measurement and store it in the buffer voltMeasBuffer and return the formatted reading, the date, and the reading elements from the buffer.
Example output:
-00.0024 mV,05/16/2014,-2.384862E-06

Also see

:FORMat[:DATA] (on page 6-38)
:READ? (on page 6-6)
[:SENSe[1]]:FUNCtion[:ON] (on page 6-70)
:TRACe:DATA? (on page 6-124)

:READ?

This query makes measurements, places them in a reading buffer, and returns the last reading.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:READ?
:READ? "<bufferName>"
:READ? "<bufferName>", <bufferElements>
```

<bufferName>	The name of the buffer where the reading is stored; if nothing is specified, defbuffer1 is used
<bufferElements>	See **Details**; if nothing is specified, READing is used

Details

This query makes the number of readings specified by `[:SENSe[1]]:COUNt`. If multiple readings are made, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command. To get multiple readings, use the `:TRACe:DATA?` command.

NOTE

To change the number of digits returned in a remote command reading, use the `:FORMat:ASCii:PRECision` command.

If you define a specific reading buffer, the reading buffer must exist before you make the measurement.

When specifying buffer elements, you can:

- Specify buffer elements in any order.
- Include any or all of the buffer elements listed below in a single list. You can repeat elements as long as the number of elements in the list is less than 14.
- Use a comma to delineate multiple elements for a data point.

The options for `<bufferElements>` are described in the following table.

Option	Description
`DATE`	The date when the data point was measured
`FORMatted`	The measured value as it appears on the front panel
`FRACtional`	The fractional seconds for the data point when the data point was measured
`READing`	The measurement reading based on the SENS:FUNC setting; if no buffer elements are defined, this option is used
`RELative`	The relative time when the data point was measured
`SEConds`	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
`SOURce`	The source value; if readback is ON, then it is the readback value, otherwise it is the programmed source value (see :SOURce[1]:<function>:READ:BACK (on page 6-87))
`SOURFORMatted`	The source value as it appears on the display
`SOURSTATus`	The status information associated with sourcing. The values returned indicate the status of the following conditions: • Overvoltage protection was active • Measured source value was read • Overtemperature condition existed • Source function level was limited • Four-wire sense was used • Output was on
`SOURUNIT`	The unit of value associated with the source value
`STATus`	The status information associated with the measurement; see the "Buffer status bits for sense measurements" table below
`TIME`	The time for the data point
`TSTamp`	The timestamp for the data point
`UNIT`	The unit of measure associated with the measurement

The output of :READ? is affected by the data format selected by :FORMat[:DATA]. If you set FORMat[:DATA] to REAL or SREAL, you will have fewer options for buffer elements. The only buffer elements available are READing and SOURce. If you request a buffer element that is not permitted for the selected data format, the instrument generates the error 1133, "Parameter 4, Syntax error, expected valid name parameters."

The STATus buffer element returns status values for the readings in the buffer. The status values are floating-point numbers that encode the status value. Refer to the following table for values.

Buffer status bits for sense measurements

Bit (hex)	Name	Decimal	Description
0x0001	STAT_QUESTIONABLE	1	Measure status questionable
0x0006	STAT_ORIGIN	6	A/D converter from which reading originated; for the Model 2450, this will always be 0 (Main)
0x0008	STAT_TERMINAL	8	Measure terminal, front is 1, rear is 0
0x0010	STAT_LIMIT2_LOW	16	Measure status limit 2 low
0x0020	STAT_LIMIT2_HIGH	32	Measure status limit 2 high
0x0040	STAT_LIMIT1_LOW	64	Measure status limit 1 low
0x0080	STAT_LIMIT1_HIGH	128	Measure status limit 1 high
0x0100	STAT_START_GROUP	256	First reading in a group

Example

```
:TRACe:MAKE "voltMeasBuffer", 10000
:SENSe:FUNCtion "VOLTage"
:COUN 10
:READ? "voltMeasBuffer", FORM, DATE, READ
:TRAC:DATA? 1, 10, "voltMeasBuffer"
```

Create a buffer named voltMeasBuffer.
Set the measurement function to voltage.
Set the count to 10.
Make the measurements and store them in the buffer voltMeasBuffer. Return the last reading as displayed on the front panel with the date, along with the unformatted reading.
Return all 10 readings from the reading buffer.
Example output is:

```
-000.06580 mV,10/14/2014,-6.580474E-05
-1.322940E-05,-7.876178E-05,-7.798489E-05,-7.201674E-05,-9.442933E-05,-7.653603E-
   06,-7.916663E-05,-8.177242E-05,-6.187183E-05,-6.580474E-05
```

Also see

:FETCh? (on page 6-1)
[:SENSe[1]]:COUNt (on page 6-69)
:TRACe:DATA? (on page 6-124)
:TRACe:TRIGger (on page 6-141)

*RCL

This command returns the instrument to the setup that was saved with the *SAV command.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
*RCL <n>
```

<n>	An integer from 0 to 4 that represents the saved setup

Details

Restores the state of the instrument from a copy of user-saved settings that are stored in the setup memory. The settings are saved using the *SAV command.

If you view the user-saved settings from the front panel of the instrument, these are stored as scripts named Setup0<n>.

Example

*RCL 3	Restores the settings stored in memory location 3.

Also see

Saving setups (on page 2-126)
*SAV (on page 6-9)

*SAV

This command saves the present instrument settings as a user-saved setup.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Nonvolatile memory	Not applicable

Usage

```
*SAV <n>
```

<n>	An integer from 0 to 4

Details

Save the present instrument settings as a user-saved setup. You can restore the settings with the *RCL command.

Any command that is affected by *RST can be saved with the *SAV command.

You can save up to 5 user-saved setups. Any settings that had been stored previously as <n> are overwritten.

If you view the user-saved setups from the front panel of the instrument, they are stored as scripts named Setup0<n>.

Example

`*SAV 2`	Saves the instrument settings in memory location 2.

Also see

Saving setups (on page 2-126)
*RCL (on page 6-9)

CALCulate subsystem

The commands in this subsystem configure and control the math and limit operations.

:CALCulate[1]:<function>:MATH:FORMat

This command specifies which math operation is performed on measurements when math operations are enabled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	PERC

Usage

```
:CALCulate[1]:<function>:MATH:FORMat <operation>
:CALCulate[1]:<function>:MATH:FORMat?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<operation>`	The name of the math operation: • y = mx+b: `MXB` • Percent: `PERCent` • Reciprocal: `RECiprocal`

Details

This specifies which math operation is performed on measurements for the selected measurement function.

You can choose one of the following math operations:

- **y = mx+b**: Manipulate normal display readings by adjusting the m and b factors.
- **Percent**: Displays measurements as the percentage of deviation from a specified reference constant.
- **Reciprocal**: The reciprocal math operation displays measurement values as reciprocals. The displayed value is `1/X`, where `X` is the measurement value (if relative offset is being used, this is the measured value with relative offset applied).

Math calculations are applied to the input signal after relative offset and before limit tests.

NOTE

If you send this command without the `<function>` parameter, it will set the state of the math format for all functions.

Example

`:CALC:VOLT:MATH:FORM MXB`
`:CALC:VOLT:MATH:MMF 0.80`
`:CALC:VOLT:MATH:MBF 50`
`:CALC:VOLT:MATH:STAT ON`
	Set the math function for voltage measurements to mx+b.
	Set the scale factor for voltage measurements to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:MBFactor (on page 6-12)
:CALCulate[1]:<function>:MATH:MMFactor (on page 6-14)
:CALCulate[1]:<function>:MATH:PERCent (on page 6-15)
:CALCulate[1]:<function>:MATH:STATe (on page 6-16)

:CALCulate[1]:<function>:MATH:MBFactor

This command specifies the offset, b, for the y = mx + b operation.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0

Usage

```
:CALCulate[1]:<function>:MATH:MBFactor <n>
:CALCulate[1]:<function>:MATH:MBFactor DEFault
:CALCulate[1]:<function>:MATH:MBFactor MINimum
:CALCulate[1]:<function>:MATH:MBFactor MAXimum
:CALCulate[1]:<function>:MATH:MBFactor?
:CALCulate[1]:<function>:MATH:MBFactor? DEFault
:CALCulate[1]:<function>:MATH:MBFactor? MINimum
:CALCulate[1]:<function>:MATH:MBFactor? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	The offset for the y = mx + b operation; the valid range is –1e12 to +1e12

Details

This attribute specifies the offset (b) for an mx + b operation.

The mx + b math operation lets you manipulate normal display readings (x) mathematically according to the following calculation:

$$y = mx + b$$

Where:

- *y* is the displayed result
- *m* is a user-defined constant for the scale factor
- *x* is the measurement reading (if you are using a relative offset, this is the measurement with relative offset applied)
- *b* is the user-defined constant for the offset factor

NOTE

If you send this command without the `<function>` parameter, it will set the scale factor for all functions.

Example

`:CALC:VOLT:MATH:FORM MXB`
`:CALC:VOLT:MATH:MMF 0.80`
`:CALC:VOLT:MATH:MBF 50`
`:CALC:VOLT:MATH:STAT ON`
	Set the math function for voltage measurements to mx+b.
	Set the scale factor for voltage measurements to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:FORMat (on page 6-10)
:CALCulate[1]:<function>:MATH:MMFactor (on page 6-14)
:CALCulate[1]:<function>:MATH:STATe (on page 6-16)

:CALCulate[1]:<function>:MATH:MMFactor

This command specifies the scale factor, m, for the y = mx + b math operation.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1.000000

Usage

```
:CALCulate[1]:<function>:MATH:MMFactor <n>
:CALCulate[1]:<function>:MATH:MMFactor DEFault
:CALCulate[1]:<function>:MATH:MMFactor MINimum
:CALCulate[1]:<function>:MATH:MMFactor MAXimum
:CALCulate[1]:<function>:MATH:MMFactor?
:CALCulate[1]:<function>:MATH:MMFactor? DEFault
:CALCulate[1]:<function>:MATH:MMFactor? MINimum
:CALCulate[1]:<function>:MATH:MMFactor? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	The scale factor; the valid range is –1e12 to +1e12

Details

This command sets the scale factor (m) for an mx + b operation for the selected measurement function.

The mx + b math operation lets you manipulate normal display readings (x) mathematically according to the following calculation:

$$y = mx + b$$

Where:

- *y* is the displayed result
- *m* is a user-defined constant for the scale factor
- *x* is the measurement reading (if you are using a relative offset, this is the measurement with relative offset applied)
- *b* is the user-defined constant for the offset factor

NOTE

If you send this command without the `<function>` parameter, it will set the scale factor for all functions.

Example

`:CALC:VOLT:MATH:FORM MXB`
`:CALC:VOLT:MATH:MMF 0.80`
`:CALC:VOLT:MATH:MBF 50`
`:CALC:VOLT:MATH:STAT ON`
	Set the math function for voltage measurements to mx+b.
	Set the scale factor for voltage measurements to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:FORMat (on page 6-10)
:CALCulate[1]:<function>:MATH:MBFactor (on page 6-12)
:CALCulate[1]:<function>:MATH:STATe (on page 6-16)

:CALCulate[1]:<function>:MATH:PERCent

This command specifies the reference constant that is used when math operations are set to percent.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1

Usage

```
:CALCulate[1]:<function>:MATH:PERCent <n>
:CALCulate[1]:<function>:MATH:PERCent DEFault
:CALCulate[1]:<function>:MATH:PERCent MINimum
:CALCulate[1]:<function>:MATH:PERCent MAXimum
:CALCulate[1]:<function>:MATH:PERCent?
:CALCulate[1]:<function>:MATH:PERCent? DEFault
:CALCulate[1]:<function>:MATH:PERCent? MINimum
:CALCulate[1]:<function>:MATH:PERCent? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	The reference used when the math operation is set to percent; the range is -1e12 to +1e12

Details

This is the constant that is used when the math operation is set to percent.

The percent math function displays measurements as percent deviation from a specified reference constant. The percent calculation is:

$$Percent = \left(\frac{input - reference}{reference}\right) \times 100\%$$

Where:

- *Percent* is the result
- *Input* is the measurement (if relative offset is being used, this is the relative offset value)
- *Reference* is the user-specified constant

NOTE

If you send this command without the `<function>` parameter, it will set the constant for all functions.

Example

`CALC:VOLT:MATH:FORM PERC`
`CALC:VOLT:MATH:PERC 50`
`CALC:VOLT:MATH:STAT ON`
	Set the math operations for voltage to percent.
	Set the percentage value to 50.
	Enable math operations.

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:FORMat (on page 6-10)
:CALCulate[1]:<function>:MATH:STATe (on page 6-16)

:CALCulate[1]:<function>:MATH:STATe

This command enables or disables math operation.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	OFF (0)

Usage

```
:CALCulate[1]:<function>:MATH:STATe <n>
:CALCulate[1]:<function>:MATH:STATe?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	Disable math operations: `OFF` or `0` Enable math operations: `ON` or `1`

Details

When this command is set to on, the math operation specified by the math format command is performed before completing a measurement.

NOTE

If you send this command without the `<function>` parameter, it sets the state of math operations for all functions.

Example

`:CALC:VOLT:MATH:FORM MXB`
`:CALC:VOLT:MATH:MMF 0.80`
`:CALC:VOLT:MATH:MBF 50`
`:CALC:VOLT:MATH:STAT ON`
	Set the math function for voltage measurements to mx+b.
	Set the scale factor for voltage measurements to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

:CALCulate[1]:<function>:MATH:FORMat (on page 6-10)
Calculations that you can apply to measurements (on page 3-6)

:CALCulate2:<function>:LIMit<Y>:AUDible

This command determines if the instrument beeper sounds when a limit test passes or fails, or disables the beeper.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	NONE

Usage

```
:CALCulate2:<function>:LIMit<Y>:AUDible <state>
:CALCulate2:<function>:LIMit<Y>:AUDible?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: `1` or `2`
`<state>`	When the beeper sounds: • Never: `NONE` • On test failure: `FAIL` • On test pass: `PASS`

Details

The tone and length of beeper cannot be adjusted.

NOTE

If you send this command without the `<function>` parameter, it will set the limit for all functions.

Example

`:CALC2:VOLT:LIM1:CLE:AUTO OFF`
`:CALC2:VOLT:LIM1:AUD FAIL`
`:CALC2:VOLT:LIM1:LOW 0.25`
`:CALC2:VOLT:LIM1:UPP 2.5`
`:CALC2:VOLT:LIMIT1:STAT ON`
`:READ?`
`:CALC2:VOLT:LIMIT1:FAIL?`
`:CALC2:VOLT:LIM1:CLE`
	Set limit autoclear off.
	Enable the beeper for limit 1 when a voltage measurement exceeds the limit.
	Set lower limit 1 for voltage to 0.25 V.
	Set upper limit 1 for voltage to 2.5 V.
	Enable limit 1 testing for voltage.
	Make a reading; the limit is checked and results display on the front panel
	Return the test results; example output if the test fails on the low limit:
	`LOW`
	Clear the test results.

Also see

:CALCulate2:<function>:LIMit<Y>:STATe (on page 6-23)

:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO

This command indicates if the test result for limit *Y* should be cleared automatically or not.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	ON (1)

Usage

```
:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO <state>
:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: `1` or `2`
`<state>`	The auto clear setting: • Disable: `OFF` or `0` • Enable: `ON` or `1`

Details

When auto clear is set to on for a measure function, limit conditions are cleared automatically after each measurement. If you are making a series of measurements, the instrument shows the limit test result of the last measurement for the pass or fail indication for the limit.

If you want to know if any of a series of measurements failed the limit, set the auto clear setting to off. When this set to off, a failed indication is not cleared automatically. It remains set until it is cleared with the clear command.

The auto clear setting affects both the high and low limits.

NOTE

If you send this command without the `<function>` parameter, it will set autoclear for all functions.

Example

`:CALC2:VOLT:LIM1:CLE:AUTO ON`
`:CALC2:VOLT:LIM1:LOW 0.25`
`:CALC2:VOLT:LIM1:UPP 2.5`
`:CALC2:VOLT:LIMIT1:STAT ON`
`:READ?`
`:CALC2:VOLT:LIMIT1:FAIL?`
	Set limit autoclear on.
	Set lower limit 1 for voltage to 0.25 V.
	Set upper limit 1 for voltage to 2.5 V.
	Enable limit 1 testing for voltage.
	Make a reading; the limit is checked and results display on the front panel
	Return the test results; example output if the test fails on the low limit:
	`LOW`
	The test results are automatically cleared.

Also see

:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate] (on page 6-19)

:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate]

This command clears the results of the limit test defined by *Y*.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate]
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: `1` or `2`

Details

Use this command to clear the test results of limit *Y* when the limit auto clear option is turned off. Both the high and low test results are cleared.

To avoid the need to manually clear the test results for a limit, turn the auto clear option on.

NOTE

If you send this command without the `<function>` parameter, it clears the limit for all functions.

Example

`:CALC2:VOLT:LIM1:CLE:AUTO OFF`
`:CALC2:VOLT:LIM1:AUD FAIL`
`:CALC2:VOLT:LIM1:LOW 0.25`
`:CALC2:VOLT:LIM1:UPP 2.5`
`:CALC2:VOLT:LIMIT1:STAT ON`
`:READ?`
`:CALC2:VOLT:LIMIT1:FAIL?`
`:CALC2:VOLT:LIM1:CLE`
	Set limit autoclear off.
	Enable the beeper for limit 1 when a voltage measurement exceeds the limit.
	Set lower limit 1 for voltage to 0.25 V.
	Set upper limit 1 for voltage to 2.5 V.
	Enable limit 1 testing for voltage.
	Make a reading; the limit is checked and results display on the front panel
	Return the test results; example output if the test fails on the low limit:
	`LOW`
	Clear the test results.

Also see

:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO (on page 6-18)
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] (on page 6-22)
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] (on page 6-24)

:CALCulate2:<function>:LIMit<Y>:FAIL?

This command queries the results of a limit test.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:CALCulate2:<function>:LIMit<Y>:FAIL?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: `1` or `2`

Details

This command queries the result of a limit test for the selected measurement function.

The response message indicates if the limit test passed or how it failed (on the high or low limit).

If autoclear is set to off, reading the results of a limit test does not clear the fail indication of the test. To clear a failure, send the clear command. To automatically clear the results, set auto clear on.

If auto clear is set to on and you are making a series of measurements, the last measurement limit determines the fail indication for the limit. If auto clear is turned off, the results return a test fail if any of one of the readings failed.

To use this attribute, you must set the limit state to on.

The results of the limit test for limit *Y*:

- `NONE`: Test passed; the measurement is between the upper and lower limits
- `HIGH`: Test failed; the measurement exceeded the upper limit
- `LOW`: Test failed; the measurement exceeded the lower limit
- `BOTH`: Test failed; the measurement exceeded both limits

Example

`:CALC2:VOLT:LIM1:CLE:AUTO OFF`	Set limit autoclear off.
`:CALC2:VOLT:LIM1:AUD FAIL`	Enable the beeper for limit 1 when a voltage measurement exceeds the limit.
`:CALC2:VOLT:LIM1:LOW 0.25`	Set lower limit 1 for voltage to 0.25 V.
`:CALC2:VOLT:LIM1:UPP 2.5`	Set upper limit 1 for voltage to 2.5 V.
`:CALC2:VOLT:LIMIT1:STAT ON`	Enable limit 1 testing for voltage.
`:READ?`	Make a reading; the limit is checked and results display on the front panel
`:CALC2:VOLT:LIMIT1:FAIL?`	Return the test results; example output if the test fails on the low limit:
`:CALC2:VOLT:LIM1:CLE`	`LOW`
	Clear the test results.

Also see

:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO (on page 6-18)
:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate] (on page 6-19)
:CALCulate2:<function>:LIMit<Y>:STATe (on page 6-23)
Limit testing and binning (on page 3-117)

:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA]

This command specifies the lower limit for limit tests.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	-1.000000E+00

Usage

```
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] <n>
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] DEFault
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] MINimum
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] MAXimum
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA]?
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA]? DEFault
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA]? MINimum
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA]? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: 1 or 2
`<n>`	The value of the lower limit (–9.99999e+11 to +9.99999e+11)

Details

This command sets the lower limit for the limit *Y* test for the selected measure function. When limit *Y* testing is enabled, this causes a fail indication to occur when the measurement value is less than this value.

NOTE

If you send this command without the `<function>` parameter, it will set the limit for all functions.

Default is 0.3 for limit 1 when the diode function is selected. The default for limit 2 for the diode function is –1.

Example

`:CALC2:VOLT:LIM1:CLE:AUTO OFF`	Set limit autoclear off.
`:CALC2:VOLT:LIM1:AUD FAIL`	Enable the beeper for limit 1 when a voltage measurement exceeds the limit.
`:CALC2:VOLT:LIM1:LOW 0.25`	Set lower limit 1 for voltage to 0.25 V.
`:CALC2:VOLT:LIM1:UPP 2.5`	Set upper limit 1 for voltage to 2.5 V.
`:CALC2:VOLT:LIMIT1:STAT ON`	Enable limit 1 testing for voltage.
`:READ?`	Make a reading; the limit is checked and results display on the front panel
`:CALC2:VOLT:LIMIT1:FAIL?`	Return the test results; example output if the test fails on the low limit: `LOW`
`:CALC2:VOLT:LIM1:CLE`	Clear the test results.

Also see

:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] (on page 6-24)

:CALCulate2:<function>:LIMit<Y>:STATe

This command enables or disables a limit test on the measurement from the selected measure function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0 (OFF)

Usage

```
:CALCulate2:<function>:LIMit<Y>:STATe <state>
:CALCulate2:<function>:LIMit<Y>:STATe?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: `1` or `2`
`<state>`	Disable the limit test: `OFF` or `0` Enable the limit test: `ON` or `1`

Details

This command enables or disables a limit test for the selected measurement function. When this attribute is enabled, the limit *Y* testing occurs on each measurement made by the instrument. Limit *Y* testing compares the measurements to the high and low limit values. If a measurement falls outside these limits, the test fails.

NOTE

If you send this command without the `<function>` parameter, it sets the state of math operations for all functions.

Example

`:CALC2:VOLT:LIM1:CLE:AUTO OFF`
`:CALC2:VOLT:LIM1:AUD FAIL`
`:CALC2:VOLT:LIM1:LOW 0.25`
`:CALC2:VOLT:LIM1:UPP 2.5`
`:CALC2:VOLT:LIMIT1:STAT ON`
`:READ?`
`:CALC2:VOLT:LIMIT1:FAIL?`
`:CALC2:VOLT:LIM1:CLE`
	Set limit autoclear off.
	Enable the beeper for limit 1 when a voltage measurement exceeds the limit.
	Set lower limit 1 for voltage to 0.25 V.
	Set upper limit 1 for voltage to 2.5 V.
	Enable limit 1 testing for voltage.
	Make a reading; the limit is checked and results display on the front panel
	Return the test results; example output if the test fails on the low limit:
	`LOW`
	Clear the test results.

Also see

:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO (on page 6-18)
:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate] (on page 6-19)
:CALCulate2:<function>:LIMit<Y>:FAIL? (on page 6-20)
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] (on page 6-22)
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] (on page 6-24)

:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA]

This command specifies the upper limit for a limit test.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1.000000E+00

Usage

```
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] <n>
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] DEFault
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] MINimum
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] MAXimum
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA]?
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA]? DEFault
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA]? MINimum
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA]? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<Y>`	Limit number: `1` or `2`
`<n>`	The value of the upper limit (–9.99999e+11 to +9.99999e+11)

Details

This command sets the high limit for the limit *Y* test for the selected measurement function. When limit *Y* testing is enabled, the instrument generates a fail indication when the measurement value is more than this value.

NOTE

If you send this command without the `<function>` parameter, it will set the limit for all functions.

Example

`:CALC2:VOLT:LIM1:CLE:AUTO OFF`	Set limit autoclear off.
`:CALC2:VOLT:LIM1:AUD FAIL`	Enable the beeper for limit 1 when a voltage measurement exceeds the limit.
`:CALC2:VOLT:LIM1:LOW 0.25`	Set lower limit 1 for voltage to 0.25 V.
`:CALC2:VOLT:LIM1:UPP 2.5`	Set upper limit 1 for voltage to 2.5 V.
`:CALC2:VOLT:LIMIT1:STAT ON`	Enable limit 1 testing for voltage.
`:READ?`	Make a reading; the limit is checked and results display on the front panel
`:CALC2:VOLT:LIMIT1:FAIL?`	Return the test results; example output if the test fails on the low limit: `LOW`
`:CALC2:VOLT:LIM1:CLE`	Clear the test results.

Also see

:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] (on page 6-22)
:CALCulate2:<function>:LIMit<Y>:STATe (on page 6-23)

DIGital subsystem

The commands in the DIGital subsystem control the digital I/O lines.

:DIGital:LINE<n>:MODE

This command sets the mode of the digital I/O line to be a digital line, trigger line, or synchronous line and sets the line to be input, output, or open-drain.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	DIG, IN

Usage

```
:DIGital:LINE<n>:MODE <lineType>, <lineDirection>
:DIGital:LINE<n>:MODE?
```

`<n>`	The digital I/O line (1 to 6)
`<lineType>`	Sets the digital line control type; the options are: • Allow direct digital control of the line: `DIGital` • Configure for trigger control: `TRIGger` • Configure as a synchronous master or acceptor: `SYNChronous`
`<lineDirection>`	Sets the line direction; the options are: • Input: `IN` • Output: `OUT` • Open drain: `OPENdrain` • Master: `MASTer` • Acceptor: `ACCeptor` See **Details** for valid combinations with line type.

Details

You can specify the line type and line direction parameters to configure each digital I/O line into one of the following modes:

- Digital open-drain, output, or input
- Trigger open-drain, output, or input
- Trigger synchronous master or acceptor

A digital line allows direct control of the digital I/O lines by writing a bit pattern to the lines. A trigger line uses the digital I/O lines to detect triggers.

Set <lineDirection> to one of the values shown in the following table.

Value	Description
IN	If the type is digital control, this automatically detects externally generated logic levels. You can read an input line, but you cannot write to it. If the type is trigger control, the line automatically responds to and detects externally generated triggers. It detects falling-edge, rising-edge, or either-edge triggers as input. This mode uses the edge setting specified by :TRIGger:DIGital<n>:IN:EDGE.
OUT	If the type is digital control, you can set the line as logic high (+5 V) or as logic low (0 V). The default level is logic low (0 V). When the instrument is in output mode, the line is actively driven high or low. If the type is trigger control, it is automatically set high or low depending on the output logic setting. Use the negative logic setting when you want to generate a falling edge trigger and use the positive logic setting when you want to generate a rising edge trigger.
OPENdrain	Configures the line to be an open-drain signal. This makes the line compatible with other instruments that use open-drain digital I/O lines or trigger signals, such as other Keithley Instruments products. If the type is digital control, the line can serve as an input, an output or both. You can read from the line or write to it. When a digital I/O line is used as an input in open-drain mode, you must write a 1 to it. If the type is trigger control, you can use the line to detect input triggers or generate output triggers. This mode uses the edge setting specified by :TRIGger:DIGital<n>:IN:EDGE.
ACCeptor	Only available with the SYNChronous trigger type. This value detects a falling-edge trigger as an input trigger and automatically latches and drives the trigger line low. Asserting the output trigger releases the latched line.
MASTer	Only available with the SYNChronous trigger type. This value detects a rising-edge trigger as an input. It asserts a TTL-low pulse for output.

Example

:DIG:LINE1:MODE DIG, OUT	Set digital I/O line 1 as a digital output line.

Also see

Digital I/O port configuration (on page 3-72)
:TRIGger:DIGital<n>:IN:EDGE (on page 6-172)

:DIGital:LINE<n>:STATe

This command sets a digital I/O line high or low when the line is set for digital control and returns the state on the digital I/O lines.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Not applicable	See **Details**

Usage

```
:DIGital:LINE<n>:STATe <state>
:DIGital:LINE<n>:STATe?
```

`<n>`	The digital I/O line (1 to 6)
`<state>`	Clear the bit (bit low): `0` Set the bit (bit high): `1`

Details

When the line mode for a digital I/O line is set to digital output (`:DIG:LINE<n>:MODE DIG, OUT`), you can set the line high or low using the `<state>` parameter. When the line mode is set to digital input (`:DIG:LINE<n>:MODE DIG, IN`), you can query the state of the digital input line.

When a reset occurs, the digital line state can be read as high because the digital line is reset to a digital input. A digital input floats high if nothing is connected to the digital line.

This returns the integer equivalent values of the binary states on all six digital I/O lines.

Set the state to zero (0) to clear the bit; set the state to one (1) to set the bit.

Example 1

`:DIG:LINE1:MODE DIG, OUT` `:DIG:LINE1:STAT 1`	Set digital I/O line 1 as a digital output line. Sets line 1 (bit B1) of the digital I/O port high.

Example 2

`:DIG:LINE1:MODE DIG, IN` `:DIG:LINE1:STAT?`	Set digital I/O line 1 as a digital input line. Query the state of line 1 on the digital I/O port. Output: `1`

Also see

Digital I/O port configuration (on page 3-72)
:DIGital:LINE<n>:MODE (on page 6-26)
:DIGital:READ? (on page 6-29)
:DIGital:WRITe <n> (on page 6-30)
:TRIGger:DIGital<n>:IN:EDGE (on page 6-172)

:DIGital:READ?

This command reads the digital I/O port.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:DIGital:READ?
```

Details

The binary equivalent of the returned value indicates the value of the input lines on the digital I/O port. The least significant bit (bit B1) of the binary number corresponds to digital I/O line 1; bit B6 corresponds to digital I/O line 6.

For example, a returned value of 42 has a binary equivalent of 101010, which indicates that lines 2, 4, 6 are high (1), and the other lines are low (0).

An instrument reset does not affect the present states of the digital I/O lines.

All six lines must be configured as digital control lines. If not, this command generates an error.

Example

`:DIG:READ?`	Assume lines 2, 4, and 6 are set high when the I/O port is read. Output: 42 This is binary 101010

Also see

Digital I/O bit weighting (on page 3-80)
Digital I/O port configuration (on page 3-72)

:DIGital:WRITe <n>

This command writes to all digital I/O lines.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:DIGital:WRITe <n>
```

<n>	The value to write to the port (0 to 63)

Details

This function writes to the digital I/O port by setting the binary state of each digital line from an integer equivalent value.

The binary representation of the value indicates the output pattern to be written to the I/O port. For example, a value of 63 has a binary equivalent of 111111 (all lines are set high); a *data* value of 42 has a binary equivalent of 101010 (lines 2, 4, and 6 are set high, and the other 3 lines are set low).

An instrument reset does not affect the present states of the digital I/O lines.

All six lines must be configured as digital control lines. If not, this command generates an error.

Example

`:DIG:WRIT 63`	Sets digital I/O lines 1 through 6 high (binary 111111).

Also see

Digital I/O bit weighting (on page 3-80)
Digital I/O port configuration (on page 3-72)

DISPlay subsystem

This subsystem contains commands that control the front-panel display.

:DISPlay:CLEar

This command clears the text from the front-panel USER swipe screen.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:DISPlay:CLEar
```

Example

```
DISP:CLE
DISP:SCR SWIPE_USER
DISP:USER1:TEXT "Batch A122"
DISP:USER2:TEXT "Test running"
```

Clear the USER swipe screen and switch to display the USER swipe screen.
Set the first line to read "Batch A122" and the second line to display "Test running".

Also see

:DISPlay:USER<n>:TEXT[:DATA] (on page 6-35)

:DISPlay:<function>:DIGits

This command determines the number of digits that are displayed for measurements on the front panel.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	5

Usage

```
:DISPlay:<function>:DIGits <n>
:DISPlay:<function>:DIGits DEFault
:DISPlay:<function>:DIGits MINimum
:DISPlay:<function>:DIGits MAXimum
:DISPlay:<function>:DIGits?
:DISPlay:<function>:DIGits? DEFault
:DISPlay:<function>:DIGits? MINimum
:DISPlay:<function>:DIGits? MAXimum
```

<function>	The measure function: • Current: CURRent[:DC] • Resistance: RESistance Voltage: VOLTage[:DC]
<n>	3.5 digit resolution: 3 4.5 digit resolution: 4 5.5 digit resolution: 5 6.5 digit resolution: 6

Details

This command affects how the reading for a measurement is displayed on the front panel of the instrument. It does not affect the number of digits returned in a remote command reading. It also does not affect the accuracy or speed of measurements.

The display digits setting is saved with the function setting, so if you use another function, then return to the function for which you set display digits, the display digits setting you set previously is retained.

The change in digits occurs the next time a measurement is made.

To change the number of digits returned in a remote command reading, use `:FORMat:ASCii:PRECision`.

NOTE

If you send this command without the `<function>` parameter, the digits values for all functions are changed.

Example

`:DISP:CURR:DIG 5`	Set the front panel to display current measurements with 5½ digits.

Also see

:FORMat:ASCii:PRECision (on page 6-36)

:DISPlay:LIGHt:STATe

This command sets the light output level of the front-panel display.

Type	Affected by	Where saved	Default value
Command and query	Power cycle	Not applicable	ON50

Usage

```
:DISPlay:LIGHt:STATe <brightness>
:DISPlay:LIGHt:STATe?
```

`<brightness>`	The brightness of the display: • Full brightness: `ON100` • 75 % brightness: `ON75` • 50 % brightness: `ON50` • 25 % brightness: `ON25` • Display off: `OFF` • Display, key lights, and all indicators off: `BLACkout`

Details

This command changes the light output of the front panel when a test requires different instrument illumination levels.

The change in illumination is temporary. The normal backlight settings are restored after a power cycle. You can use this to reset a display that is already dimmed by the front-panel Backlight Dimmer.

NOTE

Screen life is affected by how long the screen is on at full brightness. The higher the brightness setting and the longer the screen is bright, the shorter the screen life.

Example

`DISP:LIGH:STAT ON50`	Set the display brightness to 50 %.

Also see

Adjust the backlight brightness and dimmer (on page 2-11)

:DISPlay:READing:FORMat

This command determines the format that is used to display measurement readings on the front-panel display of the instrument.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	PREF

Usage

```
:DISPlay:READing:FORMat <format>
:DISPlay:READing:FORMat?
```

`<format>`	Use exponent format: `EXPonent` Add a prefix to the units symbol, such as k, m, or µ: `PREFix`

Details

This setting persists through `*RST` and power cycles.

When Prefix is selected, prefixes are added to the units symbol, such as k (kilo) or m (milli). When Exponent is selected, exponents are used instead of prefixes. When the prefix option is selected, very large or very small numbers may be displayed with exponents.

Example

`DISP:READ:FORM EXP`	Change front-panel display to show readings in exponential format.

Also see

Setting the display format (on page 2-45)

:DISPlay:SCReen

This command changes which front-panel screen is displayed.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:DISPlay:SCReen <screenName>
```

`<screenName>`	The screen to display: • Home screen: `HOME` • Home screen with large readings: `HOME_LARGe_reading` • Reading table: `READing_table` • Graph screen (opens last selected tab): `GRAPh` • Histogram screen: `HISTogram` • GRAPH swipe screen: `SWIPE_GRAPh` • SETTINGS swipe screen: `SWIPE_SETTings` • SOURCE swipe screen: `SOURce` • STATISTICS swipe screen: `SWIPE_STATistics` • USER swipe screen: `SWIPE_USER`

Example

`DISP:CLE`
`DISP:SCR SWIPE_USER`
`DISP:USER1:TEXT "Batch A122"`
`DISP:USER2:TEXT "Test running"`
	Clear and display the USER swipe screen. Set the first line to read "Batch A122" and the second line to display "Test running".

Also see

None

:DISPlay:USER<n>:TEXT[:DATA]

This command defines the text that is displayed on the front-panel USER swipe screen.

Type	Affected by	Where saved	Default value
Command only	Power cycle	Not applicable	Not applicable

Usage

```
:DISPlay:USER<n>:TEXT[:DATA] "<textMessage>"
```

`<n>`	The line of the USER swipe screen on which to display text: • Top line: `1` • Bottom line: 2
`<textMessage>`	String that contains the message; up to 20 characters for `USER1` and 32 characters for `USER2`

Details

This command defines text messages for the USER swipe screen.

If you enter too many characters, the instrument displays a warning event and shortens the message to fit.

Example

```
DISP:CLE
DISP:SCR SWIPE_USER
DISP:USER1:TEXT "Batch A122"
DISP:USER2:TEXT "Test running"
```

Clear the USER swipe screen and switch to display the USER swipe screen.
Set the first line to read "Batch A122" and the second line to display "Test running".

Also see

:DISPlay:SCReen (on page 6-34)

FORMat subsystem

The commands for this subsystem select the data format that is used to transfer instrument readings over the remote interface.

:FORMat:ASCii:PRECision

This command sets the precision (number of digits) for all numbers returned in the ASCII format.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	0

Usage

```
:FORMat:ASCii:PRECision <n>
:FORMat:ASCii:PRECision DEFault
:FORMat:ASCii:PRECision MINimum
:FORMat:ASCii:PRECision MAXimum
:FORMat:ASCii:PRECision?
:FORMat:ASCii:PRECision? DEFault
:FORMat:ASCii:PRECision? MINimum
:FORMat:ASCii:PRECision? MAXimum
```

`<n>`	The precision: • Automatic: `0` • Specific value: `1` to `16`

Details

This attribute specifies the precision (number of digits) for queries.

Note that the precision is the number of significant digits. There is always one digit to the left of the decimal point; be sure to include this digit when setting the precision.

Example

`:FORM:ASC:PREC 10`	Set a precision of 10 digits. An example of the output is: `-6.999999881E-01`

Also see

:FORMat[:DATA] (on page 6-38)

:FORMat:BORDer

This command sets the byte order for the IEEE-754 binary formats.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	SWAP

Usage

```
:FORMat:BORDer <name>
:FORMat:BORDer?
```

`<name>`	The binary byte order: • Normal byte order: `NORMal` • Reverse byte order for binary formats: `SWAPped`

Details

This attribute selected the byte order in which data is written.

The `SWAPped` byte order must be used when transmitting binary data to a computer with a Microsoft Windows operating system.

The ASCII data format can only be sent in the normal byte order. If the ASCII format is selected, the SWAPped selection is ignored.

When you select `NORMal` byte order, the data format for each element is sent as follows:

```
Byte 1 Byte 2 Byte 3 Byte 4
```

(Single precision)

When you select `SWAPped`, the data format for each element is sent as follows:

```
Byte 4 Byte 3 Byte 2 Byte 1
```

(Single precision)

The `#0` header is not affected by this command. The header is always sent at the beginning of the data string for each measurement conversion.

Example

`FORM:BORD NORM`	Use the normal byte order.

Also see

:FORMat[:DATA] (on page 6-38)

:FORMat[:DATA]

This command selects the data format that is used when transferring readings over the remote interface.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	ASC

Usage

```
:FORMat[:DATA] <type>
:FORMat[:DATA]?
```

`<type>`	The data format, which can be one of the following: • ASCII format: `ASCii` • IEEE Std. 754 double-precision format: `REAL` • IEEE Std. 754 single-precision format: `SREal`

Details

This command affects the output of `READ?`, `FETCh?`, `MEASure:<function>?`, and `TRACe:DATA?` queries over a remote interface. All other queries are returned in the ASCII format.

NOTE

The Model 2450 only responds to input commands using the ASCII format, regardless of the data format that is selected for output strings.

The IEEE Std 754 binary formats use four bytes for single-precision values and eight bytes for double-precision values.

When data is written with any of the binary formats, the response message starts with `#0` and ends with a new line. When data is written with the ASCII format, elements are separated with a comma and space.

If you set this to `REAL` or `SREAL`, you have fewer options for buffer elements with the `TRACe:DATA?`, `READ?`, `MEASURE:<function>?`, and `FETCh?` commands. The only commands available are `READing` and `SOURce`. If you request a buffer element that is not available, you see the event code 1133, "Parameter 4, Syntax error, expected valid name parameter."

Example

`FORM REAL`	Set the format to double-precision format.

Also see

:TRACe:DATA? (on page 6-124)

OUTPut subsystem

The output subsystem provides information and settings that control the output of the selected source.

:OUTPut[1]:<function>:SMODe

This command defines the state of the source when the output is turned off.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	NORM

Usage

```
:OUTPut[1]:<function>:SMODe <state>
:OUTPut[1]:<function>:SMODe?
```

`<function>`	The function to which this setting applies: • Current: `CURRent[:DC]` • Voltage: `VOLTage[:DC]`
`<state>`	The output-off setting; set to one of the following values (see the **Details** below for specifics regarding each of option): • `NORMal` • `HIMPedance` • `ZERO` • `GUARd`

Details

This command sets the state of the output when the source is off for the selected function.

When the Model 2450 is set to the normal output-off state, the following settings are made when the source is turned off:

- The measurement sense is set to 2-wire
- The voltage source is selected and set to 0 V
- The current limit is set to 10 % of the full scale of the present measurement function autorange value
- If source readback is off, Output Off is displayed in the Home screen Source area
- If source readback is on, the actual measurement is displayed in the Home screen Source area
- If measurement is set to resistance, dashes (--.----) are shown in the Home screen Source area
- The Source button on the Home screen shows the value that will be sourced when the output is turned on again

When the high-impedance output-off state is selected and the output is turned off:

- The measurement sense is set to 2-wire
- The output relay opens, disconnecting the instrument as a load

Opening the relay disconnects external circuitry from the inputs and outputs of the instrument. To prevent excessive wear on the output relay, do not use this output-off state for tests that turn the output off and on frequently.

The high-impedance output-off state should be used when the instrument is connected to a power source or another source-measure instrument. In some cases, it may also be appropriate for devices such as capacitors.

When the zero output-off state is selected and you turn off the output:

- The measurement sense is changed to 2-wire
- The voltage source is selected and set to 0
- The range is set to the presently selected range (turn off autorange)
- If the source is voltage, the current limit is not changed
- If the source is current, the current limit is set to the programmed source current value or to 10 % full scale of the present current range, whichever is greater

When the zero output-off state is selected, you can use the instrument as an ammeter because it is outputting 0 V.

When the guard output-off state is selected and the output is turned off, the following actions occur:

- The measurement sense is changed to 2-wire
- The current source is selected and set to 0 A if the source is set to current (amps); otherwise, the output remains a voltage source when the output is turned off
- The voltage limit is set to 10 % full scale of the present voltage range

Note that the front-panel display does not reflect all of the changes. For example, the 4-wire display indicator continues to display when the output is off, even though the sense is changed to 2-wire.

NOTE

If you send this command without the `<function>` parameter, it will set the output-off state for all functions.

Example

`:OUTP:CURR:SMOD HIMP`	Sets the output-off state for the current function so that the instrument opens the output relay when the output is turned off.

Also see

Output-off state (on page 2-94)
:OUTPut[1][:STATe] (on page 6-41)

:OUTPut[1]:INTerlock:TRIPped?

This command indicates that the interlock has been tripped.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:OUTPut[1]:INTerlock:TRIPped?
```

Details

This command gives you the status of the interlock. When the safety interlock signal is asserted, all voltage ranges of the instrument are available. However, when the safety interlock signal is not asserted, the 200 V range is disabled, limiting the nominal output to less than ±42 V.

When the interlock is not asserted:

- The front-panel INTERLOCK indicator is off.
- High voltage ranges are disabled.
- If you attempt to turn on the source with a voltage more than ±21 V, an event message is generated.

If `1` is returned, the interlock signal is asserted and all voltage ranges are available.

If `0` is returned, the interlock is not asserted and the 200 V range is disabled. Lower voltage ranges are available.

Example

`OUTP:INT:TRIP?`	If the interlock is not asserted, returns `0`. If the interlock is asserted, returns `1`.

Also see

None

:OUTPut[1][:STATe]

This command enables or disables the source output.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	0 (OFF)

Usage

```
:OUTPut[1][:STATe] <state>
:OUTPut[1][:STATe]?
```

`<state>`	Turn source off: `0` or `OFF` Turn source on: `1` or `ON`

Details

When the output is switched on, the instrument sources either voltage or current, as set by `:SOURce[1]:FUNCtion[:MODE]`.

Example

`:OUTP ON`	Switch the source output of the instrument to on.

Also see

:SOURce[1]:FUNCtion[:MODE] (on page 6-82)

ROUTe subsystem

The ROUTe subsystem selects which set of input and output terminals to enable (front panel or rear panel).

:ROUTe:TERMinals

This command describes which set of input and output terminals the instrument is using.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	FRON

Usage

```
:ROUTe:TERMinals <location>
:ROUTe:TERMinals?
```

`<location>`	Use the front-panel input and output terminals: `FRONt` Use the rear-panel input and output terminals: `REAR`

Details

This command selects which set of input and output terminals the instrument uses. You can select front panel or rear panel terminals.

If the output is turned on when you change from one set of terminals to the other, the output is turned off.

Example

`:ROUT:TERM REAR` `:ROUT:TERM?`	Set the instrument to use the rear-panel terminals and query to verify. Output: REAR

Also see

None

SCRipt subsystem

The SCRipt subsystem controls macro or instrument setup scripts. For additional information on macro scripts, refer to Saving front-panel settings into a macro script (on page 3-38).

SCRipt:RUN

This command runs a script.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
SCRipt:RUN "<scriptName>"
```

`<scriptName>`	The name of the script

Details

The script must be available in the instrument to be used by this command.

Example

`SCR:RUN "bufferCreate"`	Runs a script named `bufferCreate`.

Also see

Saving front-panel settings into a macro script (on page 3-38)
Scripts menu (on page 2-38)

SENSe1 subsystem

The SENSe1 subsystem commands configure and control the measurement functions of the instrument.

Many of these commands are set for a specific function (current, voltage, or resistance). For example, you can program a range setting for each function. The settings are saved with that function.

[:SENSe[1]]:<function>:AVERage:COUNt

This command sets the number of measurements that are averaged when filtering is enabled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	10

Usage

```
[:SENSe[1]]:<function>:AVERage:COUNt <n>
[:SENSe[1]]:<function>:AVERage:COUNt DEFault
[:SENSe[1]]:<function>:AVERage:COUNt MINimum
[:SENSe[1]]:<function>:AVERage:COUNt MAXimum
[:SENSe[1]]:<function>:AVERage:COUNt?
[:SENSe[1]]:<function>:AVERage:COUNt? DEFault
[:SENSe[1]]:<function>:AVERage:COUNt? MINimum
[:SENSe[1]]:<function>:AVERage:COUNt? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	The number of readings required for each filtered measurement (1 to 100)

Details

The filter count is the number of readings that are acquired and stored in the filter stack for the averaging calculation. When the filter count is larger, more filtering is done and the data is less noisy.

NOTE

If you send this command without the `<function>` parameter, it sets the filter count for all functions.

Example 1

`CURR:AVER:COUNT 10`
`CURR:AVER:TCON MOV`
`CURR:AVER ON`
	For current measurements, set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Example 2

`RES:AVER:COUNT 10`
`RES:AVER:TCON MOV`
`RES:AVER ON`
	For resistance measurements, set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Example 3

`VOLT:AVER:COUNT 10`
`VOLT:AVER:TCON MOV`
`VOLT:AVER ON`
	For voltage measurements, set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Also see

Filtering measurement data (on page 4-23)
[:SENSe[1]]:<function>:AVERage[:STATe] (on page 6-46)
[:SENSe[1]]:<function>:AVERage:TCONtrol (on page 6-47)

[:SENSe[1]]:<function>:AVERage[:STATe]

This command enables or disables the averaging filter for measurements of the selected function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	OFF (0)

Usage

```
[:SENSe[1]]:<function>:AVERage[:STATe] <state>
[:SENSe[1]]:<function>:AVERage[:STATe]?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<state>`	The filter status; set to one of the following values: • Disable the averaging filter: `OFF` or `0` • Enable the averaging filter: `ON` or `1`

Details

This command enables or disables the averaging filter. When this is enabled, the reading returned by the instrument is an averaged value, taken from multiple measurements. The settings of the filter count and filter type for the selected measure function determines how the reading is averaged.

NOTE

If you send this command without the `<function>` parameter, it sets the state of the averaging filter for all functions.

Example 1

`CURR:AVER:COUNT 10`
`CURR:AVER:TCON MOV`
`CURR:AVER ON`
	Set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Example 2

`RES:AVER:COUNT 10`
`RES:AVER:TCON MOV`
`RES:AVER ON`
	Set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Example 3

`VOLT:AVER:COUNT 10`
`VOLT:AVER:TCON MOV`
`VOLT:AVER ON`
	Set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Also see

Filtering measurement data (on page 4-23)
[:SENSe[1]]:<function>:AVERage:COUNt (on page 6-44)
[:SENSe[1]]:<function>:AVERage:TCONtrol (on page 6-47)

[:SENSe[1]]:<function>:AVERage:TCONtrol

This command sets the type of averaging filter that is used for the selected measure function when the measurement filter is enabled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	REP

Usage

```
[:SENSe[1]]:<function>:AVERage:TCONtrol <type>
[:SENSe[1]]:<function>:AVERage:TCONtrol?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<type>`	The filter type to use when filtering is enabled; set to one of the following values: • Repeating filter: `REPeat` • Moving filter: `MOVing`

Details

This command selects the type of averaging filter: Repeating average or moving average.

When the repeating average filter is selected, a set of measurements are made. These measurements are stored in a measurement stack and averaged together to produce the averaged sample. Once the averaged sample is produced, the stack is flushed and the next set of data is used to produce the next averaged sample. This type of filter is the slowest, since the stack must be completely filled before an averaged sample can be produced.

When the moving average filter is selected, the measurements are added to the stack continuously on a first-in, first-out basis. As each measurement is made, the oldest measurement is removed from the stack. A new averaged sample is produced using the new measurement and the data that is now in the stack.

NOTE

When the moving average filter is first selected, the stack is empty. When the first measurement is made, it is copied into all the stack locations to fill the stack. A true average is not produced until the stack is filled with new measurements. The size of the stack is determined by the filter count setting.

The repeating average filter produces slower results, but produces more stable results than the moving average filter. For either method, the greater the number of measurements that are averaged, the slower the averaged sample rate, but the lower the noise error. Trade-offs between speed and noise are normally required to tailor the instrumentation to your measurement application.

NOTE

If you send this command without the `<function>` parameter, it sets the filter type for all functions.

Example 1

`CURR:AVER:COUNT 10`
`CURR:AVER:TCON MOV`
`CURR:AVER ON`
	Set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Example 2

`RES:AVER:COUNT 10`
`RES:AVER:TCON MOV`
`RES:AVER ON`
	Set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Example 3

`VOLT:AVER:COUNT 10`
`VOLT:AVER:TCON MOV`
`VOLT:AVER ON`
	For voltage measurements, set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Also see

Filtering measurement data (on page 4-23)
[:SENSe[1]]:<function>:AVERage:COUNt (on page 6-44)
[:SENSe[1]]:<function>:AVERage[:STATe] (on page 6-46)

[:SENSe[1]]:<function>:AZERo[:STATe]

This command enables or disables automatic updates to the internal reference measurements (autozero) of the instrument.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	ON (1)

Usage

```
[:SENSe[1]]:<function>:AZERo[:STATe] <state>
[:SENSe[1]]:<function>:AZERo[:STATe]?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<state>`	The status of autozero: • Disable autozero: `OFF` or `0` • Enable autozero: `ON` or `1`

Details

To ensure the accuracy of readings, the instrument must periodically get new measurements of its internal ground and voltage reference. The time interval between updates to these reference measurements is determined by the integration aperture that is being used for measurements. The Model 2450 uses separate reference and zero measurements for each aperture.

By default, the instrument automatically checks these reference measurements whenever a signal measurement is made.

The time to make the reference measurements is in addition to the normal measurement time. If timing is critical, such as in sweeps, you can disable autozero to avoid this time penalty.

When autozero is set to off, the instrument may gradually drift out of specification. To minimize the drift, you can send the once command to make a reference and zero measurement immediately before a test sequence.

NOTE

If you send this command without the `<function>` parameter, it sets autozero for all functions.

Example

`VOLT:AZER OFF`	Sets autozero off for voltage measurements.

Also see

Automatic reference measurements (on page 2-117)
[:SENSe[1]]:AZERo:ONCE (on page 6-62)

[:SENSe[1]]:<function>:DELay:USER<n>

This command sets a user-defined delay that you can use in the trigger model.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list Function change	Save settings Measure configuration list	0

Usage

```
[:SENSe[1]]:<function>:DELay:USER<n> <delayTime>
[:SENSe[1]]:<function>:DELay:USER<n> DEFault
[:SENSe[1]]:<function>:DELay:USER<n> MINimum
[:SENSe[1]]:<function>:DELay:USER<n> MAXimum
[:SENSe[1]]:<function>:DELay:USER<n>?
[:SENSe[1]]:<function>:DELay:USER<n>? DEFault
[:SENSe[1]]:<function>:DELay:USER<n>? MINimum
[:SENSe[1]]:<function>:DELay:USER<n>? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	The user delay to which this time applies (1 to 5)
`<delayTime>`	The delay (0 for no delay, or 167 ns to 10 ks)

Details

To use this command in a trigger model, assign the delay to the dynamic delay block.

The delay is specific to the selected function.

Example

`:CURRent:DELay:USER1 .2` `:TRIGger:BLOCk:DELay:DYNamic 6, MEAS1`	Set user delay 1 to 0.2 s for current measurements. Set trigger block 6 to be a dynamic delay that is set to user delay 1 for the function being measured.

Also see

:TRIGger:BLOCk:DELay:DYNamic (on page 6-163)

[:SENSe[1]]:<function>:NPLCycles

This command sets the time that the input signal is measured for the selected function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1

Usage

```
[:SENSe[1]]:<function>:NPLCycles <n>
[:SENSe[1]]:<function>:NPLCycles DEFault
[:SENSe[1]]:<function>:NPLCycles MINimum
[:SENSe[1]]:<function>:NPLCycles MAXimum
[:SENSe[1]]:<function>:NPLCycles?
[:SENSe[1]]:<function>:NPLCycles? DEFault
[:SENSe[1]]:<function>:NPLCycles? MINimum
[:SENSe[1]]:<function>:NPLCycles? MAXimum
```

`<function>`	The measurement function to which this setting applies: • Current: `CURRent[:DC]` • Resistance: `RESistance` • Voltage: `VOLTage[:DC]`
`<n>`	The number of power-line cycles for each measurement: 0.01 to 10

Details

This command sets the amount of time that the input signal is measured.

The amount of time is specified as the number of power line cycles (NPLCs). Each PLC for 60 Hz is 16.67 ms (1/60) and each PLC for 50 Hz is 20 ms (1/50). For 60 Hz, if you set the NPLC to 0.1, the measure time is 1.667 ms.

This command is set for the measurement of specific functions (current, resistance, or voltage).

The shortest amount of time results in the fastest reading rate, but increases the reading noise and decreases the number of usable digits.

The longest amount of time provides the lowest reading noise and more usable digits, but has the slowest reading rate.

Settings between the fastest and slowest number of PLCs are a compromise between speed and noise.

If you change the PLCs, you may want to adjust the displayed digits to reflect the change in usable digits.

NOTE

If you send this command without the `<function>` parameter, it sets the NPLCs for all functions.

Example 1

`CURR:NPLC 0.5`	Sets the measurement time for current measurements to 0.0083 s (0.5/60).

Example 2

`RES:NPLC 0.5`	Sets the measurement time for resistance measurements to 0.0083 s (0.5/60).

Example 3

`VOLT:NPLC 0.5`	Sets the measurement time for voltage measurements to 0.0083 s (0.5/60).

Also see

Using NPLCs to adjust speed and accuracy (on page 4-8)

[:SENSe[1]]:<function>:OCOMpensated

This command enables or disables offset compensation.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	OFF (0)

Usage

```
[:SENSe[1]]:<function>:OCOMpensated <state>
[:SENSe[1]]:<function>:OCOMpensated?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<state>`	Disable offset compensation: `OFF` or `0` Enable offset compensation: `ON` or `1`

Details

The voltage offsets caused by the presence of thermoelectric EMFs (V_{EMF}) can adversely affect resistance measurement accuracy. To overcome these offset voltages, you can use offset-compensated ohms.

This feature is only applied to resistance measurements or when `[:SENSe[1]]:CURRent:UNIT` is set to `OHM`.

Example

`*RST`
`:SENS:FUNC "RES"`
`:SENS:RES:RANG:AUTO ON`
`:RES:OCOM ON`
`:COUNT 5`
`:OUTP ON`
`:TRAC:TRIG "defbuffer1"`
`:TRAC:DATA? 1, 5, "defbuffer1", SOUR, READ`
`:OUTP OFF`
	Reset the instrument.
	Set the measurement function to resistance and set the range to automatic.
	Turn offset-compensated ohms on.
	Set the measurement count to 5.
	Turn the output on.
	Make measurements and store them in `defbuffer1`.
	Retrieve readings 1 to 5 with the source value and measurement values.
	Turn the output off.

Also see

Offset-compensated ohms (on page 2-107)

[:SENSe[1]]:<function>:RANGe:AUTO

This command determines if the measurement range is set manually or automatically for the selected measure function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	ON (1)

Usage

```
[:SENSe[1]]:<function>:RANGe:AUTO <state>
[:SENSe[1]]:<function>:RANGe:AUTO?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<state>`	Set the measurement range manually: `OFF` or `0` Set the measurement range automatically: `ON` or `1`

Details

Auto range selects the best range in which to measure the signal that is applied to the input terminals of the instrument. When auto range is enabled, the range increases at 120 percent of range and decreases occurs when the reading is <10 percent of nominal range.

This command determines how the range is selected.

When this command is set to off, you must set the range. If you do not set the range, the instrument remains at the range that was selected by autorange.

When this command is set to on, the instrument automatically goes to the most sensitive range to perform the measurement.

If a range is manually selected through the front panel or a remote command, this command is automatically set to off.

Example

`RES:RANG:AUTO ON`	Set the range to be selected automatically for resistance measurements.

Also see

[:SENSe[1]]:<function>:RANGe[:UPPer] (on page 6-56)
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit (on page 6-54)

[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit

This command selects the lower limit for measurements of the selected function when the range is selected automatically.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	Current: 10e-9 Voltage: 20 Resistance: 20

Usage

```
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit <n>
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit?
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit? DEFault
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit? MINimum
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit? MAXimum
```

`<function>`	The measurement function to which this setting applies: • Current: `CURRent[:DC]` • Resistance: `RESistance` • Voltage: `VOLTage[:DC]`
`<n>`	The lower limit: • Current: 1e-8 to 1 A • Resistance: 2 to 200e6 Ω • Voltage: 0.02 to 200 V

Details

You can use this command when automatic range selection is enabled. It prevents the instrument from selecting a range that is below this limit. Because the lowest ranges generally require longer settling times, setting the low limit that is appropriate for your application but above the lowest possible range can make measurements require less settling time.

The lower limit must be less than the upper limit.

Though you can send any value when you send this command, the instrument selects the next highest range value. For example, if you send 15 for the lowest voltage range, the instrument will be set to the 20 V range as the low limit.

Example

`:VOLT:RANG:AUTO:LLIM 15` `:VOLT:RANG:AUTO:LLIM?`	Set the low range for voltage measurements to 20 V. Output: `2.000000E+01`

Also see

Ranges (on page 2-112)
[:SENSe[1]]:<function>:RANGe:AUTO (on page 6-53)

[:SENSe[1]]:<function>:RANGe:AUTO:ULIMit

When autorange is selected, this command represents the highest measurement range that is used when the instrument selects the measurement range automatically.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	Resistance: 200e6

Usage

```
[:SENSe[1]]:<function>:RANGe:AUTO:ULIMit <n>
[:SENSe[1]]:<function>:RANGe:AUTO:ULIMit?
[:SENSe[1]]:<function>:RANGe:AUTO:ULIMit? DEFault
[:SENSe[1]]:<function>:RANGe:AUTO:ULIMit? MINimum
[:SENSe[1]]:<function>:RANGe:AUTO:ULIMit? MAXimum
```

`<function>`	The measurement function to which this setting applies: • Current (query only): `CURRent[:DC]` • Resistance: `RESistance` • Voltage (query only): `VOLTage[:DC]`
`<n>`	The upper limit: • Current: 1e-8 to 1 A • Resistance: 2 to 200e6 Ω • Voltage: 0.02 to 200 V

Details

For the resistance function, you can use this command when automatic range selection is enabled to put an upper bound on the range that is used for resistance measurements.

The upper limit must be more than the lower limit.

If the lower limit is equal to the upper limit, automatic range setting is effectively disabled.

Example

`:SENSe:RESistance:RANGe:AUTO:ULIMit 20`	Set the upper limit to 20 Ω.

Also see

[:SENSe[1]]:<function>:RANGe:AUTO (on page 6-53)
[:SENSe[1]]:<function>:RANGe:AUTO:LLIMit (on page 6-54)

[:SENSe[1]]:<function>:RANGe[:UPPer]

This command determines the positive full-scale measure range.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	Current: 1e-04 Resistance: 200,000 Voltage: 2e-02

Usage

```
[:SENSe[1]]:<function>:RANGe[:UPPer] <n>
[:SENSe[1]]:<function>:RANGe[:UPPer] DEFault
[:SENSe[1]]:<function>:RANGe[:UPPer] MINimum
[:SENSe[1]]:<function>:RANGe[:UPPer] MAXimum
[:SENSe[1]]:<function>:RANGe[:UPPer]?
[:SENSe[1]]:<function>:RANGe[:UPPer]? DEFault
[:SENSe[1]]:<function>:RANGe[:UPPer]? MINimum
[:SENSe[1]]:<function>:RANGe[:UPPer]? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	Set this command to a specific value or a preset value: • Current: 10 nA to 1 A • Resistance: 20 Ω to 200 MΩ • Voltage: 0.02 V to 200 V

Details

You can assign any real number using this command. The instrument selects the closest fixed range that is large enough to measure the entered number. For example, for current measurements, if you expect a reading of approximately 9 mA, set the range to 9 mA to select the 10 mA range. When you read this setting, you see the positive full-scale value of the measurement range that the instrument is presently using.

This command is primarily intended to eliminate the time that is required by the instrument to automatically search for a range.

When a range is fixed, any signal greater than the entered range generates an overrange condition. When an overrange condition occurs, the front panel displays "Overflow" and the remote interface returns `9.9e+37`.

If the source function is the same as the measurement function (for example, sourcing voltage and measuring voltage), the measurement range is the same as the source range, regardless of measurement range setting. However, the setting for the measure range is retained, and when the source function is changed (for example, from sourcing voltage to sourcing current), the retained measurement range is used.

If you change the range while the output is off, the instrument does not update the hardware settings, but if you read the range setting, the return is the setting that will be used when the output is turned on. If you set a range while the output is on, the new setting takes effect immediately.

NOTE

When you set a value for the measurement range, the measurement autorange setting is automatically disabled for the selected measurement function.

The range for measure functions defaults to autorange. When you switch from autorange to range, the range is set to the last selected autorange value.

Example 1

`:SENS:CURR:RANG 10E-6`	Select the 10 µA range.

Example 2

`:SENS:RES:RANG 2E6`	Select the 2 MΩ range.

Example 3

`:SENS:VOLT:RANG 50e-3`	Select the 200 mV range.

Also see

Ranges (on page 2-112)
[:SENSe[1]]:<function>:RANGe:AUTO (on page 6-53)

[:SENSe[1]]:<function>:RELative

This command contains the relative offset value.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0

Usage

```
[:SENSe[1]]:<function>:RELative <n>
[:SENSe[1]]:<function>:RELative?
[:SENSe[1]]:<function>:RELative? DEFault
[:SENSe[1]]:<function>:RELative? MINimum
[:SENSe[1]]:<function>:RELative? MAXimum
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<n>`	The relative offset value: • Current: –1.05 to 1.05 • Resistance: –210e6 to 210e6 Voltage: –210 to 210

Details

This command specifies the relative offset value that can be applied to new measurements. When relative offset is enabled, all subsequent measured readings are offset by the value that is set for this command.

You can set this value, or have the instrument acquire a value. If the instrument acquires the value, read this setting to return the value that was measured internally.

Example

`CURR:REL .5` `CURR:REL:STAT ON`	Set the relative offset for current measurements to 0.5. Enable relative offset.

Also see

Relative offset (on page 3-3)
[:SENSe[1]]:<function>:RELative:ACQuire (on page 6-59)
[:SENSe[1]]:<function>:RELative:STATe (on page 6-60)

[:SENSe[1]]:<function>:RELative:ACQuire

This command acquires a measurement and stores it as the relative offset value.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:<function>:RELative:ACQuire
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`

Details

This command triggers the instrument to make a new measurement for the selected function. This measurement is then stored as the new relative offset level.

When you send this command, the instrument does not apply any math, limit test, or filter settings to the measurement, even if they are set. It is a measurement that is made as if these settings are disabled.

If an error event occurs during the measurement, `nil` is returned and the relative offset level remains at the last valid setting.

You must change to the function for which you want to acquire a value before sending this command.

The instrument must have relative offset enabled to use the acquired relative offset value.

After executing this command, you can use the `[:SENSe[1]]:<function>:RELative?` command to return the last relative level value that was acquired or set.

Example

`FUNC "RES"`
`RES:REL:ACQ`
`RES:REL?`
`RES:REL:STAT ON`
	Switch to resistance measurements. Acquire a relative offset value for resistance measurements.
	Query for the offset value.
	Turn relative offset on.
	Example output:
	`-5.4017E-10`

Also see

[:SENSe[1]]:<function>:RELative (on page 6-58)
[:SENSe[1]]:<function>:RELative:STATe (on page 6-60)

[:SENSe[1]]:<function>:RELative:STATe

This command enables or disables the application of a relative offset value to the measurement.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0 (OFF)

Usage

```
[:SENSe[1]]:<function>:RELative:STATe <state>
[:SENSe[1]]:<function>:RELative:STATe?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<state>`	Disable the relative offset: `OFF` or `0` Enable the relative offset: `ON` or `1`

Details

When relative measurements are enabled, all subsequent measured readings are offset by the relative offset value. You can enter a relative offset value or have the instrument acquire a relative offset value.

Each returned measured relative reading is the result of the following calculation:

Displayed reading = Actual measured reading - Relative offset value

Example

`:SENS:FUNC "VOLT"`
`:SENS:VOLT:REL 5`
`:SENSe:VOLT:REL:STATe ON`
	Set the measurement function to volts with a relative offset of 5 V and enable the relative offset function.

Also see

Relative offset (on page 3-3)
[:SENSe[1]]:<function>:RELative (on page 6-58)
[:SENSe[1]]:<function>:RELative:ACQuire (on page 6-59)

[:SENSe[1]]:<function>:RSENse

This command selects local (2-wire) or remote (4-wire) sensing.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0 (OFF)

Usage

```
[:SENSe[1]]:<function>:RSENse <state>
[:SENSe[1]]:<function>:RSENse?
```

`<function>`	The measurement function to which this setting applies: • Current: `CURRent[:DC]` • Resistance: `RESistance` • Voltage: `VOLTage[:DC]`
`<state>`	Disable remote sensing (2-wire): `0` or `OFF` Enable remote sensing (4-wire): `1` or `ON`

Details

This command determines if 2-wire (local) or 4-wire (remote) sensing is used.

When you use 4-wire sensing, voltages are measured at the device under test (DUT). For the source voltage, if the sensed voltage is lower than the programmed amplitude, the voltage source increases the voltage until the sensed voltage is the same as the programmed amplitude. This compensates for IR drop in the output test leads.

Using 4-wire sensing with voltage measurements eliminates any voltage drops that may be in the test leads between the Model 2450 and the DUT.

When you are using 2-wire sensing, voltage is measured at the output connectors.

When you are measuring resistance, you can enable 4-wire sensing to make 4-wire resistance measurements.

When the output is off, 4-wire sensing is disabled and the instrument uses 2-wire sense, regardless of the sense setting. When the output is on, the selected sense setting is used.

Example

`VOLT:RSEN ON`	Set the remote sense for voltage measurements.

Also see

Two-wire local sense connections (on page 2-87)
Four-wire remote sense connections (on page 2-89)

[:SENSe[1]]:<function>:UNIT

This command sets the units of measurement that are displayed on the front panel of the instrument and stored in the reading buffer.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	Current: AMP Voltage: VOLT

Usage

```
[:SENSe[1]]:<function>:UNIT <unitOfMeasure>
[:SENSe[1]]:<function>:UNIT?
```

`<function>`	The measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` Voltage: `VOLTage[:DC]`
`<unitOfMeasure>`	Current: `OHM`, `WATT`, or `AMP` Voltage: `OHM`, `WATT`, or `VOLT`

Details

The change in measurement units is displayed when the next measurement occurs.

Example

`VOLT:UNIT WATT`	Changes the front-panel display and buffer readings for voltage measurements to be displayed as power readings in watts.

Also see

None

[:SENSe[1]]:AZERo:ONCE

This command causes the instrument to refresh the reference and zero measurements once.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:AZERo:ONCE
```

Details

This command forces a refresh of the reference and zero measurements that are used for the present aperture setting for the selected function.

When autozero is set to off, the instrument may gradually drift out of specification. To minimize the drift, you can send the once command to make a reference and zero measurement immediately before a test sequence.

If the NPLC setting is less than 0.2 PLC, sending autozero once can result in delay of more than a second.

Example

`FUNC "VOLT"` `AZER:ONCE`	Do a one-time refresh of the reference and zero measurements for the voltage function.

Also see

Automatic reference measurements (on page 2-117)
[:SENSe[1]]:<function>:AZERo[:STATe] (on page 6-48)

[:SENSe[1]]:CONFiguration:LIST:CATalog?

This command returns the name of one measure configuration list that is stored on the instrument.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:CATalog?
```

Details

You can use this command to retrieve the names of measure configuration lists that are stored in the instrument.

This command returns one name each time you send it. This command returns an empty string when there are no more names to return. If the command returns an empty string the first time you send it, no measure configuration lists have been created for the instrument.

Example

`CONF:LIST:CAT?`	Send this command to retrieve the name of one measure configuration list. To get all stored lists, send it again until it returns an empty string.

Also see

Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-64)

[:SENSe[1]]:CONFiguration:LIST:CREate

This command creates an empty measure configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:CREate "<name>"
```

`<name>`	A string that represents the name of a measure configuration list

Details

This command creates an empty configuration list. To add configuration indexes to this list, you need to use the store command.

Configuration lists are not saved when the instrument is turned off. To save a configuration list, use a saved setup to store the instrument settings, which include defined configuration lists.

Example

```
:SENS:CONF:LIST:CRE "MyMeasList"
```

Creates a measure configuration list named `MyMeasList`.

Also see

*SAV (on page 6-9)
Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:STORe (on page 6-68)

[:SENSe[1]]:CONFiguration:LIST:DELete

This command deletes a measure configuration list.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:DELete "<name>"
[:SENSe[1]]:CONFiguration:LIST:DELete "<name>", <index>
```

`<name>`	A string that represents the name of a measure configuration list
`<index>`	A number that defines a specific configuration index in the configuration list

Details

Deletes a configuration list. If the index is not specified, the entire configuration list is deleted. If the index is specified, only the specified configuration index in the list is deleted.

When an index is deleted from a configuration list, the index numbers of the following indexes are shifted up by one. For example, if you have a configuration list with 10 indexes and you delete index 3, the index that was numbered 4 becomes index 3, and the all the following indexes are renumbered in sequence to index 9. Because of this, if you want to delete several nonconsecutive indexes in a configuration list, it is best to delete the higher numbered index first, then the next lower index, and so on. This also means that if you want to delete all the indexes in a configuration list, you must delete index 1 repeatedly until all indexes have been removed.

Example

`:SENSe:CONF:LIST:DELete "myMeasList"`	Deletes a configuration list named `myMeasList`.
`:SENSe:CONF:LIST:DELete "myMeasList", 2`	Deletes configuration index 2 in a configuration list named `myMeasList`.

Also see

Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-64)

[:SENSe[1]]:CONFiguration:LIST:QUERy?

This command returns a list of TSP commands and parameter settings that are stored in the specified configuration index.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:QUERy? "<name>", <index>
[:SENSe[1]]:CONFiguration:LIST:QUERy? "<name>", <index>, <fieldSeparator>
```

`<name>`	A string that represents the name of a measure configuration list
`<index>`	A number that defines a specific configuration index in the configuration list
`<fieldSeparator>`	A separator for the data: • Comma (default): `1` • Semicolon: 2 • New line: 3

Details

This command returns data for one configuration index.

For additional information about the information this command returns, see Instrument settings stored in a measure configuration list (on page 3-42).

Example

```
:SENS:CONF:LIST:QUER? "MyMeasList", 2, 3
```

Returns the TSP commands and parameter settings that represent the settings in configuration index 2.

Also see

Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-64)
TSP command reference (on page 8-1)

[:SENSe[1]]:CONFiguration:LIST:RECall

This command recalls a configuration index in a measure configuration list.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:RECall "<name>"
[:SENSe[1]]:CONFiguration:LIST:RECall "<name>", <index>
```

`<name>`	A string that represents the name of a measure configuration list
`<index>`	A number that defines a specific configuration index in the configuration list

Details

Use this command to recall the settings stored in a specific configuration index in a specific configuration list. If you do not specify an index when you send the command, it recalls the settings stored in the first configuration index in the specified configuration list.

If you recall an invalid index (for example, calling index 3 when there are only two indexes in the configuration list) or try to recall an index from an empty configuration list, event code 2790, "Configuration list, error, does not exist" is displayed.

Each index contains the settings for the selected function. Settings for other functions are not affected when the configuration list index is recalled.

NOTE

Recall source configuration lists before measure configuration lists. This order ensures that dependencies between source and measure settings will be properly handled.

This command returns data for one configuration index.

For additional information about the information this command returns, see Instrument settings stored in a measure configuration list (on page 3-42).

Example

`:SENSe:CONF:LIST:RECall "MyMeasList", 5`	Recalls configuration index 5 in a configuration list named `MyMeasList`.
`:SENSe:CONF:LIST:RECall "MyMeasList"`	Since an index was not specified, this command recalls configuration index 1 from a configuration list named `MyMeasList`.

Also see

Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-64)
[:SENSe[1]]:CONFiguration:LIST:STORe (on page 6-68)

[:SENSe[1]]:CONFiguration:LIST:SIZE?

This command returns the size (number of configuration indexes) of a measure configuration list.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:SIZE? "<name>"
```

`<name>`	A string that represents the name of a measure configuration list

Details

This command returns the size (number of configuration indexes) of a measure configuration list. The size of the list is equal to the number of configuration indexes in a configuration list.

Example

`:SENSe:CONF:LIST:SIZE? "MyMeasList"`	Returns the number of configuration indexes in a measure configuration list named `MyMeasList`. Example output: 3

Also see

Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-64)

[:SENSe[1]]:CONFiguration:LIST:STORe

This command stores the active measure settings into the named configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Saved settings	Not applicable

Usage

```
[:SENSe[1]]:CONFiguration:LIST:STORe "<name>"
[:SENSe[1]]:CONFiguration:LIST:STORe "<name>", <index>
```

`<name>`	A string that represents the name of a measure configuration list
`<index>`	A number that defines a specific configuration index in the configuration list

Details

Use this command to store the active settings to a configuration index in a configuration list. If you do not include the `<index>` parameter, the configuration index is appended to the end of the list.

Refer to Instrument settings stored in a measure configuration list (on page 3-42) for a complete list of measure settings that the instrument stores.

Example

`:SENSe:CONF:LIST:STOR "MyConfigList"`	Stores the active settings of the instrument to the end of the configuration list named `MyConfigList`.
`:SENSe:CONF:LIST:STOR "MyConfigList", 5`	Stores the active settings of the instrument to the configuration list named `MyConfigList` in configuration index 5.

Also see

Configuration lists (on page 3-39)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-64)

[:SENSe[1]]:COUNt

This command sets the number of measurements to make when a measurement is requested.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	1

Usage

```
[:SENSe[1]]:COUNt <n>
[:SENSe[1]]:COUNt DEFault
[:SENSe[1]]:COUNt MINimum
[:SENSe[1]]:COUNt MAXimum
[:SENSe[1]]:COUNt?
[:SENSe[1]]:COUNt? DEFault
[:SENSe[1]]:COUNt? MINimum
[:SENSe[1]]:COUNt? MAXimum
```

<n>	The number of measurements (1 to 300,000)

Details

This command sets the number of measurements that are made when a measurement is requested. This command does not affect the trigger model.

This command sets the count for all measure functions.

If you set the count to a value that is larger than the capacity of the reading buffer and the buffer fill mode is set to continuous, the buffer wraps until the number of readings specified have occurred. The earliest readings in the count are overwritten. If the buffer is set to fill once, readings stop when the buffer is filled, even if the count is not complete.

NOTE

To get better performance from the instrument, use the Simple Loop trigger model template instead of using the count command.

Example

`:TRAC:CLEAR`
`:COUN 10`
`:MEAS?`
`:TRAC:DATA? 1,10`
	Clear data from the reading buffer.
	Set the count to 10.
	Make ten measurements.
	Returns the last measurement.
	Example output:
	`-5.693831E-05`
	Read all ten measurements.
	Example output:
	`-7.681046E-05,-2.200288E-04,-9.086048E-05,-6.388056E-05,-7.212282E-05,-4.874761E-05,-4.741654E-04,-6.811028E-05,-5.110232E-05,-5.693831E-05`

Also see

:MEASure? (on page 6-4)
:TRACe:DATA? (on page 6-124)
:TRIGger:LOAD "SimpleLoop" (on page 6-192)

[:SENSe[1]]:FUNCtion[:ON]

This command selects the active measure function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	CURR

Usage

```
[:SENSe[1]]:FUNCtion[:ON] "<function>"
[:SENSe[1]]:FUNCtion[:ON]?
```

`<function>`	A string that contains the measure function: • Current: `CURRent[:DC]` • Resistance: `RESistance` • Voltage: `VOLTage[:DC]`

Details

Set this command to the type of measurement you want to make.

Reading this command returns the measure function that is presently active.

Example

`:FUNC "VOLTage"`	Make the voltage measurement function the active function.

Also see

Making resistance measurements (on page 2-102)
Source and measure using SCPI commands (on page 2-108)

SOURce subsystem

The commands in the SOURce subsystem configure and control the current source and voltage source.

:SOURce[1]:CONFiguration:LIST:CATalog?

This command returns the name of one source configuration list.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:CONFiguration:LIST:CATalog?
```

Details

You can use this command to see the names of source configuration lists stored on the instrument.

This command returns one name each time you send it. This command returns an empty string if there are no more names to return. If the command returns an empty string the first time you send it, no source configuration lists have been created for the instrument.

Example

`:SOUR:CONF:LIST:CAT?`	Send this command to return the name of one source configuration list stored on the instrument. To get all stored configuration lists, resend this command until it returns an empty string.

Also see

Configuration lists (on page 3-39)
:SOURce[1]:CONFiguration:LIST:CREate (on page 6-71)

:SOURce[1]:CONFiguration:LIST:CREate

This command creates an empty source configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:CONFiguration:LIST:CREate "<name>"
```

`<name>`	A string that represents the name of a source configuration list

Details

This command creates an empty configuration list. To add configuration indexes to this list, you need to use the store command.

Configuration lists are not saved when the instrument is turned off. If you want to save a configuration list, use a saved setup to store the instrument settings, which include defined configuration lists.

Example

```
:SOUR:CONF:LIST:CRE "MySourceList"
```

Creates a source configuration list named `MySourceList`.

Also see

Configuration lists (on page 3-39)
*SAV (on page 6-9)
:SOURce[1]:CONFiguration:LIST:STORe (on page 6-75)

:SOURce[1]:CONFiguration:LIST:DELete

This command deletes a source configuration list.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:CONFiguration:LIST:DELete "<name>"
:SOURce[1]:CONFiguration:LIST:DELete "<name>", <index>
```

`<name>`	A string that represents the name of a source configuration list
`<index>`	A number that defines a specific configuration index in the configuration list

Details

Deletes a configuration list. If the index is not specified, the entire configuration list is deleted. If the index is specified, only the specified configuration index in the list is deleted.

When an index is deleted from a configuration list, the index numbers of the following indexes are shifted up by one. For example, if you have a configuration list with 10 indexes and you delete index 3, the index that was numbered 4 becomes index 3, and the all the following indexes are renumbered in sequence to index 9. Because of this, if you want to delete several nonconsecutive indexes in a configuration list, it is best to delete the higher numbered index first, then the next lower index, and so on. This also means that if you want to delete all the indexes in a configuration list, you must delete index 1 repeatedly until all indexes have been removed.

Example

`:SOURce:CONF:LIST:DEL "MySourceList"`	Deletes a configuration list named `MySourceList`.
`:SOURce:CONF:LIST:DEL "MySourceList", 2`	Deletes configuration index 2 in the configuration list named `MySourceList`.

Also see

Configuration lists (on page 3-39)
:SOURce[1]:CONFiguration:LIST:CREate (on page 6-71)

:SOURce[1]:CONFiguration:LIST:QUERy?

This command returns a list of TSP commands and parameter settings that are stored in the specified configuration index.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:CONFiguration:LIST:QUERy? "<name>", <point>
:SOURce[1]:CONFiguration:LIST:QUERy? "<name>", <point>, <fieldSeparator>
```

`<name>`	A string that represents the name of a source configuration list
`<point>`	A number that defines a specific configuration index in the configuration list
`<fieldSeparator>`	A separator for the data: • Comma (default): `1` • Semicolon: 2 • New line: 3

Details

This command can only return data for one configuration index. To get data for additional configuration indexes, resend the command and specify different configuration indexes.

Refer to Instrument settings stored in a source configuration list (on page 3-44) for a complete list of source settings that the instrument stores in a source configuration list.

Example

```
:SOUR:CONF:LIST:QUER? "MySourceList", 2
```

Returns the TSP commands and parameter settings that represent the settings in configuration index 2.

Also see

Configuration lists (on page 3-39)
:SOURce[1]:CONFiguration:LIST:CREate (on page 6-71)
Instrument settings stored in a source configuration list (on page 3-44)

:SOURce[1]:CONFiguration:LIST:RECall

This command recalls a specific configuration index in a specific source configuration list.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:CONFiguration:LIST:RECall "<name>", <index>
```

`<name>`	A string that represents the name of a source configuration list
`<index>`	A number that defines a specific configuration index in the configuration list

Details

Use this command to recall the settings stored in a specific configuration index in a specific configuration list. If you do not specify an index when you send the command, it recalls the settings stored in the first configuration index in the specified configuration list.

If you recall an invalid index (for example, calling index 3 when there are only two indexes in the configuration list) or try to recall an index from an empty configuration list, event code 2790, "Configuration list, error, does not exist" is displayed.

Each index contains the settings for the selected function. Settings for other functions are not affected when the configuration list index is recalled.

NOTE

Recall source configuration lists before measure configuration lists. This order ensures that dependencies between source and measure settings will be properly handled.

Example

`:SOURce:CONF:LIST:REC "MySourceList", 5`	Recalls configuration index 5 in a configuration list named `MySourceList`.
`:SOURce:CONF:LIST:RECall "MySourceList"`	Since an index was not specified, this command recalls configuration index 1 from a configuration list named `MySourceList`.

Also see

Configuration lists (on page 3-39)
:SOURce[1]:CONFiguration:LIST:CREate (on page 6-71)

:SOURce[1]:CONFiguration:LIST:SIZE?

This command returns the number of configuration indexes of a source configuration list.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Query

```
:SOURce[1]:CONFiguration:LIST:SIZE? "<name>"
```

`<name>`	A string that represents the name of a source configuration list

Details

The size of the list is equal to the number of configuration indexes in a configuration list.

Example

`:SOUR:CONF:LIST:SIZE? "MySourceList"`	Returns the number of configuration indexes in a source configuration list named `MySourceList`.

Also see

Configuration lists (on page 3-39)
:SOURce[1]:CONFiguration:LIST:CREate (on page 6-71)

:SOURce[1]:CONFiguration:LIST:STORe

This command stores the active source settings into the named configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle Source configuration list	Saved settings	Not applicable

Usage

```
:SOURce[1]:CONFiguration:LIST:STORe "<name>", <index>
```

`<name>`	A string that represents the name of a source configuration list
`<index>`	A number that defines a specific configuration index in the configuration list

Details

Use this command to store the active source settings to a configuration index in a configuration list. If the index is defined, the configuration list is stored in that index. If the index is not defined, the configuration index is appended to the end of the list. If a configuration index already exists for the specified index, the new configuration overwrites the existing configuration index.

Refer to Instrument settings stored in a source configuration list (on page 3-44) for information about the settings this command stores.

Example

`:SOURce:CONF:LIST:CRE "biasLevel"`
`:SOURce:FUNC VOLT`
`:SOURce:VOLT:LEV 5`
`:SOURce:CONF:LIST:STORE "biasLevel"`
	Create a configuration list named `biasLevel`.
	Set the source function to voltage and the source voltage level to 5 V.
	Store the configuration list and append it to the end of the `biasLevel` configuration list.

Also see

:SOURce[1]:CONFiguration:LIST:CREate (on page 6-71)

:SOURce[1]:<function>:DELay

This command contains the source delay.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	Not applicable

Usage

```
:SOURce[1]:<function>:DELay <n>
:SOURce[1]:<function>:DELay?
:SOURce[1]:<function>:DELay? DEFault
:SOURce[1]:<function>:DELay? MINimum
:SOURce[1]:<function>:DELay? MAXimum
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<n>`	The delay in seconds (0 to 4)

Details

This command sets a delay for the selected source function. This delay is in addition to normal settling times.

After the programmed source is turned on, this delay allows the source level to settle before a measurement is made.

If you set a specific source delay (`smu.source.delay`), source autodelay is turned off.

When source autodelay is turned on, the manual source delay setting is overwritten with the autodelay setting.

When either a source delay or autodelay is set, the delay is applied to the first source output and then only when the magnitude of the source changes.

NOTE

If you send this command without the `<function>` parameter, it sets the delay for all functions.

Example

`SOUR:VOLT:DEL DEF`	Set the delay for the voltage source to the default value.

Also see

:SOURce[1]:<function>:DELay:AUTO (on page 6-77)

:SOURce[1]:<function>:DELay:AUTO

This command enables or disables the automatic delay that occurs when the source is turned on.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	1 (ON)

Usage

```
:SOURce[1]:<function>:DELay:AUTO <state>
:SOURce[1]:<function>:DELay:AUTO?
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<state>`	Disable the source auto delay: `OFF` or `0` Enable the source auto delay: `ON` or `1`

Details

When autodelay is turned on, the actual delay that is set depends on the range.

When source autodelay is on, if you set a source delay, the autodelay is turned off.

When source autodelay is on, the manual source delay setting is overwritten with the autodelay setting.

The delay is applied to the first source output and then only when the magnitude of the source changes.

Example

`SOUR:CURR:DEL:AUTO OFF`	Turn off auto delay when current is being sourced.

Also see

:SOURce[1]:<function>:DELay (on page 6-76)

:SOURce[1]:<function>:DELay:USER<n>

This command sets a user-defined delay that you can use in the trigger model.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	0.000000E+00

Usage

```
:SOURce[1]:<function>:DELay:USER<n> <delayTime>
:SOURce[1]:<function>:DELay:USER<n>?
:SOURce[1]:<function>:DELay:USER<n>? DEFault
:SOURce[1]:<function>:DELay:USER<n>? MINimum
:SOURce[1]:<function>:DELay:USER<n>? MAXimum
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<n>`	The number that identifies this user delay (1 to 5)
`<delayTime>`	The time of the delay in seconds (0 to 10,000)

Details

To use this command in a trigger model, assign the delay to the dynamic delay block.

The delay is specific to the selected function.

Example

`:SOUR:VOLT:DEL:USER1 5`	Set user delay for source 1 to 5 s.
`:TRIG:BLOC:SOUR:STAT 1, ON`	Set trigger block 1 to turn the source output on.
`:TRIG:BLOC:DEL:DYN 2, SOUR1`	Set trigger block 2 to a dynamic delay that calls source user delay 1.
`:TRIG:BLOC:MEAS 3`	Set trigger block 3 to make a measurement.
`:TRIG:BLOC:SOUR:STAT 4, OFF`	Set trigger block 4 to turn the source output off.
`:TRIG:BLOC:BRAN:COUN 5, 10, 1`	Set trigger block 5 to branch to block 1 ten times.
`:INIT`	Start the trigger model.

Also see

:TRIGger:BLOCk:DELay:DYNamic (on page 6-163)

:SOURce[1]:<function>:HIGH:CAPacitance

This command enables or disables high-capacitance mode.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	0 (OFF)

Usage

```
:SOURce[1]:<function>:HIGH:CAPacitance <state>
:SOURce[1]:<function>:HIGH:CAPacitance?
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<state>`	Turn high capacitance off: `OFF` or `0` Turn high capacitance on: `ON` or `1`

Details

When the instrument is measuring low current and is driving a capacitive load, you may see overshoot, ringing, and instability. You can enable the high capacitance mode to minimize these problems.

The settings for high-capacitance mode apply when you operate the instrument using the 1 µA and above current ranges. When operating using the 1 A range, the high-capacitance setting will not affect the instrument rise time or current measure settling time.

Use this command with limited autorange (low) with the low range set to 1 µA.

Example

`SOUR:CURR:HIGH:CAP ON`	Turn the high capacitance mode on when sourcing current.

Also see

High-capacitance operation (on page 4-21)

:SOURce[1]:<function>[:LEVel][:IMMediate][:AMPLitude]

This command immediately selects a fixed amplitude for the selected source function.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	0

Usage

```
:SOURce[1]:<function>[:LEVel][:IMMediate][:AMPLitude] <n>
:SOURce[1]:<function>[:LEVel][:IMMediate][:AMPLitude]?
:SOURce[1]:<function>[:LEVel][:IMMediate][:AMPLitude]? DEFault
:SOURce[1]:<function>[:LEVel][:IMMediate][:AMPLitude]? MINimum
:SOURce[1]:<function>[:LEVel][:IMMediate][:AMPLitude]? MAXimum
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<n>`	Current: –1.05 A to 1.05 A Voltage: –210 V to 210 V

Details

This command sets the output level of the voltage or current source. If the output is on, the new level is sourced immediately.

The sign of the source level dictates the polarity of the source. Positive values generate positive voltage or current from the high terminal of the source relative to the low terminal. Negative values generate negative voltage or current from the high terminal of the source relative to the low terminal.

If a manual source range is selected, the level cannot exceed the specified range. For example, if the voltage source is on the 2 V range (auto range is disabled), you cannot set the voltage source amplitude to 3 V. When auto range is selected, the amplitude can be set to any level.

Example

`SOUR:FUNC VOLT` `SOUR:VOLT 1`	Set the instrument to source voltage and set it to source 1 V.

Also see

:SOURce[1]:<function>:RANGe (on page 6-85)
:SOURce[1]:<function>:RANGe:AUTO (on page 6-86)

:SOURce[1]:<function>:<x>LIMit[:LEVel]

This command selects the source limit for measurements of the selected function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	Voltage: 105 µA Current: 21 V

Usage

```
:SOURce[1]:CURRent:VLIMit[:LEVel] <n>
:SOURce[1]:CURRent:VLIMit[:LEVel]?
:SOURce[1]:CURRent:VLIMit[:LEVel]? DEFault
:SOURce[1]:CURRent:VLIMit[:LEVel]? MINimum
:SOURce[1]:CURRent:VLIMit[:LEVel]? MAXimum
:SOURce[1]:VOLTage:ILIMit[:LEVel] <n>
:SOURce[1]:VOLTage:ILIMit[:LEVel]?
:SOURce[1]:VOLTage:ILIMit[:LEVel]? DEFault
:SOURce[1]:VOLTage:ILIMit[:LEVel]? MINimum
:SOURce[1]:VOLTage:ILIMit[:LEVel]? MAXimum
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<n>`	The limit: • Current: 1 nA to 1.05 A • Voltage: 0.02 V to 210 V

Details

This command sets the source limit for measurements. The Model 2450 cannot source levels that exceed this limit.

The values that can be set for this command are limited by the setting for the overvoltage protection limit.

This value can also be limited by the measurement range. If a specific measurement range is set, the limit must be more than 0.1 % of the measurement range. If you set the measurement range to be automatically selected, the measurement range does not affect the limit.

If you change the source range to a level that is not appropriate for this limit, the instrument changes the source limit to a limit that is appropriate to the range and a warning is generated.

Limits are absolute values.

Example

`:SOUR:CURR:VLIM 15`	Set the voltage limit to 15 V.

Also see

:SOURce[1]:<function>:PROTection[:LEVel] (on page 6-83)
:SOURce[1]:<function>:<x>LIMit[:LEVel]:TRIPped? (on page 6-82)

:SOURce[1]:<function>:<x>LIMit[:LEVel]:TRIPped?

This command indicates if the source exceeded the limits that were set for the selected measurements.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:CURRent:VLIMit[:LEVel]:TRIPped?
:SOURce[1]:VOLTage:ILIMit[:LEVel]:TRIPped?
```

Details

You can use this command check the limit state of the source.

If the limits were exceeded, the instrument clamps the source to keep the source within the set limits.

If the source did not exceed the set limits, the return is `0`. If the source did exceed the set limits, the return is `1`.

Example 1

`SOUR:CURR:VLIM:TRIP?`	Returns a value that indicates whether or not the source exceeded the current limits.

Example 2

`SOUR:VOLT:ILIM:TRIP?`	Return value indicates whether or not the source has exceeded the voltage limits.

Also see

:SOURce[1]:<function>:<x>LIMit[:LEVel] (on page 6-81)

:SOURce[1]:FUNCtion[:MODE]

This command contains the source function, which can be voltage or current.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	`VOLT`

Usage

```
:SOURce[1]:FUNCtion[:MODE] <function>
:SOURce[1]:FUNCtion[:MODE]?
```

`<function>`	Voltage source function: `VOLTage` Current source function: `CURRent`

Details

When you set this command, it configures the instrument as either a voltage source or a current source.

Example

`SOUR:FUNC CURR` `SOUR:FUNC?`	Set the source function of the instrument to be a current source and query the source function. Output: `CURR`

Also see

None

:SOURce[1]:<function>:PROTection[:LEVel]

This command sets the overvoltage protection setting of the source output.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	NONE

Usage

```
:SOURce[1]:VOLTage:PROTection[:LEVel] <n>
:SOURce[1]:VOLTage:PROTection[:LEVel]?
```

`<n>`	The overvoltage protection level, set as `<n>`, where `<n>` is `PROT2`, `PROT5`, `PROT10`, `PROT20`, `PROT40`, `PROT60`, `PROT80`, `PROT100`, `PROT120`, `PROT140`, `PROT160`, `PROT180`, or `NONE`

Details

Overvoltage protection restricts the maximum voltage level that the instrument can source. It is in effect when either current or voltage is sourced.

This protection is in effect for both positive and negative output voltages.

When this attribute is used in a test sequence, it should be set before turning the source on.

⚠ WARNING

Even with the overvoltage protection set to the lowest value (2 V), never touch anything connected to the terminals of the Model 2450 when the output is on. Always assume that a hazardous voltage (greater than 30 V_{RMS}) is present when the output is on. To prevent damage to the device under test or external circuitry, do not set the voltage source to levels that exceed the value that is set for overvoltage protection.

Example

`sour:volt:prot prot40` `sour:volt:prot?`	Set the voltage source protection to 40 V and query the value. The output is: `PROT40`

Also see

Overvoltage protection (on page 2-110)

:SOURce[1]:<function>:PROTection[:LEVel]:TRIPped?

This command indicates if the overvoltage source protection feature is active.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:VOLTage:PROTection[:LEVel]:TRIPped?
```

Details

When overvoltage protection is active, the instrument restricts the maximum voltage level that the instrument can source.

If the voltage source does not exceed the set limits, the return is `0`. If the voltage source exceeds the set limits, the return is `1`.

Example

`SOUR:VOLT:PROT:TRIP?`	If overvoltage protection is active, the output is: `1`

Also see

Overvoltage protection (on page 2-110)
:SOURce[1]:<function>:PROTection[:LEVel] (on page 6-83)

:SOURce[1]:<function>:RANGe

This command selects the range for the source for the selected source function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	Current: 1e-08 Voltage: 2e-02

Usage

```
:SOURce[1]:<function>:RANGe <n>
:SOURce[1]:<function>:RANGe?
:SOURce[1]:<function>:RANGe? DEFault
:SOURce[1]:<function>:RANGe? MINimum
:SOURce[1]:<function>:RANGe? MAXimum
```

`<function>`	The source function to which this setting applies: • Current: `CURRent` • Voltage: `VOLTage`
`<n>`	Set to the maximum expected voltage or current to be sourced; the ranges are: • Current: –1 A to 1 A • Voltage: –200 V to 200 V

Details

This command manually selects the measurement range for the specified source.

If you select a specific source range, the range must be large enough to source the value. If not, an overrange condition can occur.

If an overrange condition occurs, an event is displayed and the change to the setting is ignored.

The fixed current source ranges are 10 nA, 100 nA, 1 µA, 10 µA, 100 µA, 1 mA, 10 mA, 100 mA, and 1 A.

The fixed voltage source ranges are 20 mV, 200 mV, 2 V, 20 V, and 200 V.

When you read this value, the instrument returns the positive full-scale value that the instrument is presently using.

This command is intended to eliminate the time required by the automatic range selection.

To select the range, you can specify the approximate source value that you will use. The instrument selects the lowest range that can accommodate that level. For example, if you expect to source levels around 50 mV, send 0.05 (or 50e-3) to select the 200 mV range.

NOTE

If automatic range selection is set to on, when you select a specific range, automatic is set to off. To set the range to automatic selection, use the source autorange command.

Example

`:SOURce:VOLTage:RANGe 3`	Send this command to source levels around 3 V. This example selects the 20 V range for the voltage source.

Also see

:SOURce[1]:<function>:RANGe:AUTO (on page 6-86)
Ranges (on page 2-112)

:SOURce[1]:<function>:RANGe:AUTO

This command determines if the range is selected manually or automatically for the selected source function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	1 (ON)

Usage

```
:SOURce[1]:CURRent:RANGe:AUTO <state>
:SOURce[1]:CURRent:RANGe:AUTO?
:SOURce[1]:VOLTage:RANGe:AUTO <state>
:SOURce[1]:VOLTage:RANGe:AUTO?
```

`<state>`	Disable automatic source range: `0` or `OFF` Enable automatic source range: `1` or `ON`

Details

This command indicates the state of the range for the selected source. When automatic source range is disabled, the source range is set manually.

When automatic source range is enabled, the instrument selects the range that is most appropriate for the value that is being sourced. The output level controls the range. If you read the range after the output level is set, the instrument returns the range that the instrument chose as appropriate for that source level.

If the source range is set to a specific value from the front panel or a remote command, the setting for automatic range is set to disabled.

Example

`SOUR:CURR:RANG:AUTO ON`	Enable the automatic source range.

Also see

:SOURce[1]:<function>:RANGe (on page 6-85)

:SOURce[1]:<function>:READ:BACK

This command determines if the instrument records the measured source value or the configured source value when making a measurement.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Source configuration list	Save settings Source configuration list	1 (ON)

Usage

```
:SOURce[1]:VOLTage:READ:BACK <state>
:SOURce[1]:VOLTage:READ:BACK?
:SOURce[1]:CURRent:READ:BACK <state>
:SOURce[1]:CURRent:READ:BACK?
```

`<state>`	Disable read back: `0` or `OFF` Enable read back: `1` or `ON`

Details

When source readback is off, the instrument records and displays the source value you set. When you use the actual source value (source readback on), the instrument measures the actual source value immediately before making the device under test measurement.

Using source readback results in more accurate measurements, but also a reduction in measurement speed.

When source readback is on, the front-panel display shows the measured source value and the buffer records the measured source value immediately before the device-under-test measurement. When source readback is off, the front-panel display shows the configured source value and the buffer records the configured source value immediately before the device-under-test measurement.

Example

`*RST`	Reset the instrument to default settings.
`TRAC:MAKE "MyBuffer", 100`	Make a buffer named "MyBuffer" that can hold 100 readings.
`SOUR:FUNC VOLT`	Set source function to voltage.
`SENS:FUNC "CURR"`	Set the measurement function to current.
`SOUR:VOLT:READ:BACK ON`	Set read back on.
`SOUR:VOLT 10`	Set the instrument to take 100 readings.
`COUNT 100`	Turn the output on.
`OUTP ON`	Take a measurement (100 readings).
`READ? "MyBuffer"`	Turn the output off.
`OUTP OFF`	Get the source values and measurements from the buffer.
`TRAC:DATA? 1, 100, "MyBuffer", SOUR, READ`	

Also see

:SOURce[1]:FUNCtion[:MODE] (on page 6-82)

:SOURce[1]:LIST:<function>

This command allows you to set up a list of custom values for a sweep.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:LIST:CURRent <list>
:SOURce[1]:LIST:CURRent?
:SOURce[1]:LIST:VOLTage <list>
:SOURce[1]:LIST:VOLTage?
```

`<list>`	Current: –1.05 A to 1.05 A Voltage: –210 V to 210 V See **Details**

Details

This command defines a list of up to 100 source values for a source list. This list is used by the `:SOURce[1]:SWEep:<function>:LIST` command to define the source values for the sweep.

When you start the sweep, the instrument sequentially sources each current or voltage value in the list. A measurement is performed at each source level.

If there is an existing list, it is replaced by the new list.

When you send this command, the instrument creates a source configuration list named `CurrCustomSweepList` if the function is set to current or `VoltCustomSweepList` if the function is set to voltage.

To add source values to an existing list, use the `:SOURce[1]:LIST:<function>:APPend` command.

Example

```
*RST
SENS:FUNC "CURR"
SENS:CURR:RANG:AUTO ON
SENS:CURR:RSEN OFF
SOUR:FUNC VOLT
SOUR:VOLT:RANG 20
SOUR:VOLT:ILIM 1
SOUR:LIST:VOLT 1, 5, 1, 5, 1, 5
SOUR:SWE:VOLT:LIST 1, 0.2
INIT
*WAI

TRAC:DATA? 1, 6, "defbuffer1", SOUR, READ
```

This example will source 1 V, 5 V, 1 V, 5 V, 1 V, 5 V and measure the resulting current at each voltage point. The time duration of each voltage point is 200 ms.

Also see

:SOURce[1]:LIST:<function>:APPend (on page 6-89)
:SOURce[1]:SWEep:<function>:LIST (on page 6-95)

:SOURce[1]:LIST:<function>:APPend

This command adds values to the source list for the selected source function.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:LIST:CURRent:APPend <list>
:SOURce[1]:LIST:VOLTage:APPend <list>
```

`<list>`	Current: –1.05 to 1.05 A Voltage: –210 to 210 V

Details

This adds up to 100 values to the list created with `:SOURce[1]:LIST:<function>`. The new values are added to the end of the existing values. You can have a total of 2500 values in a list, but you must append them in groups of 100.

If the list does not exist, this command creates one.

Example

```
*RST
SENS:FUNC "CURR"
SENS:CURR:RANG:AUTO ON
SENS:CURR:RSEN OFF
SOUR:FUNC VOLT
SOUR:VOLT:RANG 20
SOUR:VOLT:ILIM 1
SOUR:LIST:VOLT 1, 5, 1, 5, 1, 5
SOUR:LIST:VOLT:APP 1, 5, 1, 5, 1, 5
SOUR:SWE:VOLT:LIST 1, 0.2
INIT
*WAI

TRAC:DATA? 1, 12, "defbuffer1", SOUR, READ
```

This example will create a source configuration list (`VoltCustomSweepList`) and source 1 V, 5 V six times and measure the resulting current at each voltage point. The duration of each voltage point is 200 ms.

Also see

:SOURce[1]:LIST:<function> (on page 6-88)

:SOURce[1]:LIST:<function>:POINts?

This command queries the length of the source list for the selected source function.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SOURce[1]:LIST:CURRent:POINts?
:SOURce[1]:LIST:VOLTage:POINts?
```

Details

This command returns the length of the specified source list. The response message indicates the number of source values in the list.

Example

```
*RST
SENS:FUNC "CURR"
SENS:CURR:RANG:AUTO ON
SENS:CURR:RSEN OFF
SOUR:FUNC VOLT
SOUR:VOLT:RANG 20
SOUR:VOLT:ILIM 1
SOUR:LIST:VOLT 1, 5, 1, 5, 1, 5
SOUR:SWE:VOLT:LIST 1, 0.2
INIT
*WAI

TRAC:DATA? 1, 6, "defbuffer1", SOUR, READ
SOUR:LIST:VOLT:POIN?
```

This example will source 1 V, 5 V, 1 V, 5 V, 1 V, 5 V and measure the resulting current at each voltage point. The time duration of each voltage point is 200 ms.
Check the number of points in the list.
Output:
6

Also see

:SOURce[1]:LIST:<function> (on page 6-88)
:SOURce[1]:LIST:<function>:APPend (on page 6-89)

:SOURce[1]:SWEep:<function>:LINear

This command sets up a linear sweep for a fixed number of measurement points.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>, <delay>
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>, <delay>, <count>
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>
:SOURce[1]:SWEep:<function>:LINear <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>, "<bufferName>"
```

`<function>`	Voltage sweep: `VOLTage` Current sweep: `CURRent`
`<start>`	The voltage or current source level at which the sweep starts: • Current: -1.05 A to 1.05 A • Voltage: -210 V to 210 V
`<stop>`	The voltage or current at which the sweep stops: • Current: -1.05 A to 1.05 A • Voltage: -210 V to 210 V
`<points>`	The number of source-measure points between the start and stop values of the sweep (2 to 1e6); to calculate the number of source-measure points in a sweep, use the following formula: Points = [(Stop - Start) / Step] + 1
`<delay>`	The delay between measurement points; default is `-1`, which enables autodelay, or a specific delay value from 50 µs to 10,000 s, or 0 for no delay
`<count>`	The number of times to run the sweep; default is `1`: • Infinite loop: `0` • Finite loop: `1` to `268435455`
`<rangeType>`	The source range that is used for the sweep: • Most sensitive source range for each source level in the sweep: `AUTO` • Best fixed range: `BEST` (default) • Present source range for the entire sweep: `FIXed`
`<failAbort>`	Abort the sweep if the source limit is exceeded: `ON` (default) Complete the sweep if the source limit is exceeded: `OFF`
`<dual>`	Determines if the sweep runs from start to stop and then from stop to start: • Sweep from start to stop only: `OFF` (default) • Sweep from start to stop, then stop to start: `ON`
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

When the sweep is started, the instrument sources a specific voltage or current value to the device under test (DUT). A measurement is made for each point of the sweep.

When the sweep command is sent, it clears any existing trigger models, creates a source configuration list, and populates the trigger model. To run the sweep, initiate the trigger model.

The sweep continues until the source outputs the specified stop level. At this level, the instrument performs another measurement and then stops the sweep.

When you specify a delay, a delay block is added to the sweep trigger model. This delay is added to any source delay you may have set. For example, if you set 10 ms for the source delay and 25 ms for the sweep delay, the actual delay is 35 ms.

The range type specifies the source range that is used for the sweep. You can select the following options:

- Auto: The instrument automatically goes to the most sensitive source range for each source level in the sweep.
- Best fixed: The instrument selects a single fixed source range that accommodates all the source levels in the sweep. This avoids overshoots during sweeps.
- Fixed: The source remains on the range that is set when the sweep is started. If a sweep point that exceeds the capability of the source range, the source outputs the maximum level for that range.

Example

`*RST`	Reset the instrument to its defaults.
`SOUR:FUNC VOLT`	Set the source function to voltage.
`SOUR:VOLT:RANG 20`	Set the source range to 20 V. Set the measure function to current with a range of 100 µA.
`SENS:FUNC "CURR"`	
`SENS:CURR:RANG 100e-6`	
`SOUR:SWE:VOLT:LIN 0, 10, 20, 1e-3, 1, FIXED`	Set up a linear sweep that sweeps from 0 to 10 V in 20 points with a source delay of 1 ms, a sweep count of 1, and a fixed source range.
`INIT`	Start the sweep.

Also see

Sweep operation (on page 3-60)

:SOURce[1]:SWEep:<function>:LINear:STEP

This command sets up a linear source sweep configuration list and trigger model with a fixed number of steps.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>, <delay>
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>, <delay>, <count>
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>, <delay>, <count>,
   <rangeType>
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>, <delay>, <count>,
   <rangeType>, <failAbort>
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>
:SOURce[1]:SWEep:<function>:LINear:STEP <start>, <stop>, <steps>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>, "<bufferName>"
```

<function>	Voltage sweep: VOLTage Current sweep: CURRent
<start>	The voltage or current source level at which the sweep starts: • Current: -1.05 A to 1.05 A • Voltage: -210 V to 210 V
<stop>	The voltage or current at which the sweep stops: • Current: -1.05 A to 1.05 A • Voltage: -210 V to 210 V
<steps>	The step size at which the source level will change; step size must be greater than 0
<delay>	The delay between measurement points; default is -1, which enables autodelay, or a specific delay value from 50 µs to 10,000 s; or 0 for no delay
<count>	The number of times to run the sweep; default is 1: • Infinite loop: 0 • Finite loop: 1 to 268435455
<rangeType>	The source range that is used for the sweep: • Most sensitive source range for each source level in the sweep: AUTO • Best fixed range: BEST (default) • Present source range for the entire sweep: FIXed
<failAbort>	Abort the sweep if the source limit is exceeded: ON (default) Complete the sweep if the source limit is exceeded: OFF
<dual>	Determines if the sweep runs from start to stop and then from stop to start: • Sweep from start to stop only: OFF (default) • Sweep from start to stop, then stop to start: ON
<bufferName>	A string that indicates the reading buffer; the default buffers (defbuffer1 or defbuffer2) or the name of a user-defined buffer; if no buffer is specified, defbuffer1 is used

Detail

When the sweep is started, the instrument sources a specific voltage or current voltage to the device under test (DUT). A measurement is made for each point of the sweep.

When the sweep command is sent, it deletes the existing trigger model and creates a trigger model with a uniform series of ascending or descending voltage or current changes, called steps. To run the sweep, initiate the trigger model.

The sweep continues until the source outputs the stop level, which is calculated from the number of steps. A measurement is performed at each source step (including the start and stop levels). At this level, the instrument performs another measurement and then stops the sweep.

The instrument uses the step size parameter to determine the number of source level changes. The source level changes in equal steps from the start level to the stop level. To avoid a setting conflicts error, make sure the step size is greater than the start value and less than the stop value. To calculate the number of source-measure points in a sweep, use the following formula:

$$step = \frac{stop - start}{points - 1}$$

When you specify a delay, a delay block is added to the sweep trigger model. This delay is added to any source delay you may have set. For example, if you set 10 ms for the source delay and 25 ms for the delay in the for the log sweep command, the actual delay is 35 ms.

The range type specifies the source range that is used for the sweep. You can select the following options:

- Auto: The instrument automatically goes to the most sensitive source range for each source level in the sweep.
- Best fixed: The instrument selects a single fixed source range that accommodates all the source levels in the sweep. This avoids overshoots during sweeps.
- Fixed: The source remains on the range that is set when the sweep is started. If a sweep point that exceeds the capability of the source range, the source outputs the maximum level for that range.

Example

```
*RST
SOUR:FUNC CURR
SOUR:CURR:RANGE 1
SENS:FUNC "VOLT"
SENS:VOLT:RANGE 20
SOUR:SWE:CURR:LIN:STEP -1.05, 1.05, .25, 10e-3, 1, FIXED
INIT
```

Reset the instrument to its defaults.
Set the source function to current.
Set the source range to 1 A. Set the measure function to voltage with a range of 20 V.
Set up a linear step sweep that sweeps from -1.05 A to 1.05 A in 0.25 A increments with a source delay of 1 ms, a sweep count of 1, and a fixed source range. The name of the configuration list that is created for this sweep is `CurrLinearSweep`.
Start the sweep.

Also see

Sweep operation (on page 3-60)

:SOURce[1]:SWEep:<function>:LIST

This command sets up a sweep based on a configuration list, which allows you to customize the sweep.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:SWEep:<function>:LIST <startIndex>
:SOURce[1]:SWEep:<function>:LIST <startIndex>, <delay>
:SOURce[1]:SWEep:<function>:LIST <startIndex>, <delay>, <count>
:SOURce[1]:SWEep:<function>:LIST <startIndex>, <delay>, <count>, <failAbort>
:SOURce[1]:SWEep:<function>:LIST <startIndex>, <delay>, <count>, <failAbort>,
   "<bufferName>"
:SOURce[1]:SWEep:<function>:LIST <startIndex>, <delay>, <count>, <failAbort>,
   "<bufferName>", "<configListName>"
```

`<function>`	The source function: • Current: `CURRent` • Voltage: `VOLTage`
`<startIndex>`	The index in the configuration list where the sweep starts; default is `1`
`<delay>`	The delay between measurement points; default is 0 for no delay or you can set a specific delay value from 50 µs to 10,000 s
`<count>`	The number of times to run the sweep; default is `1`: • Infinite loop: `0` • Finite loop: `1` to `268435455`
`<failAbort>`	Determines if the sweep is stopped immediately if a limit is exceeded; options are: • Abort the sweep if a limit is exceeded: `ON` • Complete the sweep even if a limit is exceeded: `OFF`
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<configListName>`	The name of the source configuration list that the sweep uses; this must be defined before sending this command

Details

This command allows you to set up a custom sweep, using a configuration list to specify the source levels.

When you specify a delay, a delay block is added to the sweep trigger model. This delay is added to any source delay you may have set. For example, if you set 10 ms for the source delay and 25 ms for the delay in the for the log sweep command, the actual delay is 35 ms.

A configuration list must be created before you send this command. You can either use the `:SOURce[1]:LIST:<function>` to set up the configuration list, or you can create your own configuration list.

To run the sweep, initiate the trigger model.

Example 1

```
*RST
SENS:FUNC "CURR"
SENS:CURR:RANG:AUTO ON
SENS:CURR:RSEN OFF
SOUR:FUNC VOLT
SOUR:VOLT:RANG 20
SOUR:VOLT:ILIM 1
SOUR:LIST:VOLT 1, 5, 1, 5, 1, 5
SOUR:SWE:VOLT:LIST 1, 0.2
INIT
*WAI

TRAC:DATA? 1, 6, "defbuffer1", SOUR, READ
```

This example uses the `:SOURce[1]:LIST:<function>` command to set up the configuration list that is used by the sweep.
This example will source 1 V, 5 V, 1 V, 5 V, 1 V, 5 V and measure the resulting current at each voltage point. The time duration of each voltage point is 200 ms.

Example 2

```
SOUR:CONF:LIST:CRE "biasLevel"
SOUR:FUNC VOLT
SENS:FUNC "CURR"
SOUR:VOLT:LEV 5
SOUR:CONF:LIST:STORE "biasLevel"
SOUR:SWE:VOLT:LIST 1, .001, 1, 1, "defbuffer2", "biasLevel"
INIT
```

This example uses a user-defined configuration list.
Create a configuration list named `biasLevel`. Set the source function to 5 V and the measure function to current.
Store the configuration list.
Set up a voltage sweep that uses the configuration list, starting at index point 1 with a delay of 1 ms. The sweep is to abort if the source limit is exceeded, store data in `defbuffer2`, and use the configuration list `biasLevel`.

Also see

Configuration lists (on page 3-39)
:INITiate[:IMMediate] (on page 6-146)
:SOURce[1]:LIST:<function> (on page 6-88)
Sweep operation (on page 3-60)

:SOURce[1]:SWEep:<function>:LOG

This command sets up a logarithmic sweep for a set number of measurement points.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>, <count>
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>, "<bufferName>"
:SOURce[1]:SWEep:<function>:LOG <start>, <stop>, <points>, <delay>, <count>,
   <rangeType>, <failAbort>, <dual>, "<bufferName>", <asymptote>
```

<function>	The source function: • CURRent • VOLTage
<start>	The voltage or current source level at which the sweep starts: • Current: 1 pA to 1.05 A • Voltage: 1 pV to 210 V
<stop>	The voltage or current at which the sweep stops: • Current: 1 pA to 1.05 A • Voltage: 1 pV to 210 V
<points>	The number of source-measure points between the start and stop values of the sweep (2 to 1e6); to calculate the number of source-measure points in a sweep, use the following formula: Points = [(Stop - Start) / Step] + 1
<delay>	The delay between measurement points; default is -1, which enables autodelay, or a specific delay value from 50 µs to 10,000 s, or 0 for no delay
<count>	The number of times to run the sweep; default is 1: • Infinite loop: 0 • Finite loop: 1 to 268435455
<rangeType>	The source range that is used for the sweep: • Most sensitive source range for each source level in the sweep: AUTO • Best fixed range: BEST (default) • Present source range for the entire sweep: FIXed
<failAbort>	Abort the sweep if the source limit is exceeded: ON (default) Complete the sweep if the source limit is exceeded: OFF
<dual>	Determines if the sweep runs from start to stop and then from stop to start: • Sweep from start to stop only: OFF (default) • Sweep from start to stop, then stop to start: ON

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<asymptote>`	Default is 0; see **Details**

Details

When the sweep is started, the instrument sources a specific voltage or current value to the device under test (DUT). A measurement is made for each point of the sweep.

When the sweep command is sent, it clears the existing trigger model and creates a new trigger model. To run the sweep, initiate the trigger model.

The sweep continues until the source outputs the specified stop level. At this level, the instrument performs another measurement and then stops the sweep.

When you specify a delay, a delay block is added to the sweep trigger model. This delay is added to any source delay you may have set. For example, if you set 10 ms for the source delay and 25 ms for the delay in the for the log sweep command, the actual delay is 35 ms.

The range type specifies the source range that is used for the sweep. You can select the following options:

- Auto: The instrument automatically goes to the most sensitive source range for each source level in the sweep.
- Best fixed: The instrument selects a single fixed source range that accommodates all the source levels in the sweep. This avoids overshoots during sweeps.
- Fixed: The source remains on the range that is set when the sweep is started. If a sweep point that exceeds the capability of the source range, the source outputs the maximum level for that range.

The asymptote changes the inflection of the sweep curve and allows it to sweep through zero. You can use the asymptote parameter to customize the inflection and offset of the source value curve. Setting this parameter to zero provides a conventional logarithmic sweep. The asymptote value is the value that the curve has at either positive or negative infinity, depending on the direction of the sweep. The asymptote value must not be equal to or between the starting and ending values. It must be outside the range defined by the starting and ending values.

A configuration list must be created before you send this command. You can either use the `:SOURce[1]:LIST:<function>` to set up the configuration list, or you can create your own configuration list.

Example

```
*RST
SOUR:FUNC VOLT
SOUR:VOLT:RANG 20
SENS:FUNC "CURR"
SENS:CURR:RANG 100e-6
SOUR:SWE:VOLT:LOG .1, 10, 20, 1e-3, 1, FIXED
INIT
```

Reset the instrument to its defaults.
Set the source function to voltage.
Set the source range to 20 V.
Set the measure function to current.
Set the current range to 100 µA.
Set up a log sweep that sweeps from 0.1 to 10 V in 20 steps with a source delay of 1 ms, a sweep count of 1, and a fixed source range.
Start the sweep.

Also see

:INITiate[:IMMediate] (on page 6-146)
Sweep operation (on page 3-60)

STATus subsystem

The STATus subsystem controls the status registers of the instrument. For additional information on the status model, see Status model (on page C-1).

:STATus:CLEar

This function clears event registers and the event log.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:STATus:CLEar
```

Details

This command clears the event registers of the Questionable Event and Operation Event Register set. It does not affect the Questionable Event Enable or Operation Event Enable registers.

Example

`:STATus:CLEar`	Clear the bits in the registers

Also see

*CLS (on page B-2)

:STATus:OPERation:CONDition?

This command reads the Operation Event Register of the status model.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

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

Details

This command reads the contents of the Operation Condition Register, which is one of the Operation Event Registers.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-16).

Example

`:STAT:OPER:COND?`	Returns the contents of the Operation Condition Register.

Also see

Operation Event Register (on page C-8)

:STATus:OPERation:ENABle

This command sets or reads the contents of the Operation Event Enable Register of the status model.

Type	Affected by	Where saved	Default value
Command and query	STATus:PRESet	Not applicable	0

Usage

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

<n>	The status of the operation status register

Details

This command sets or reads the contents of the Enable register of the Operation Event Register.

When one of these bits is set, when the corresponding bit in the Operation Event Register or Operation Condition Register is set, the OSB bit in the Status Byte Register is set.

When sending binary values, preface <n> with #b. When sending hexadecimal values, preface <n> with #h. No preface is needed when sending decimal values.

Example

`:STAT:OPER:ENAB #b0101000000000000`	Sets the 12 and 14 bits of the operation status enable register using a decimal value.
	You could also send the decimal value:
	`:STAT:OPER:ENAB 20480`
	Or the hexadecimal value:
	`:STAT:OPER:ENAB #h5000`

Also see

Operation Event Register (on page C-8)

:STATus:OPERation[:EVENt]?

This command reads the Operation Event Register of the status model.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:STATus:OPERation[:EVENt]?
```

Details

This attribute reads the operation event register of the status model.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Example

`stat:oper?`	Returns the contents of the Operation Event Register of the status model.

Also see

Operation Event Register (on page C-8)

:STATus:OPERation:MAP

This command allows you to map event numbers to bits in the Operation Event Registers.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Not applicable	Not applicable

Usage

```
:STATus:OPERation:MAP <bitNumber>, <setEvent>
:STATus:OPERation:MAP <bitNumber>, <setEvent>, <clearEvent>
:STATus:OPERation:MAP? <bitNumber>
```

bitNumber	The bit number that is mapped to an event (0 to 14)
setEvent	The number of the event that sets the bits in the condition and event registers; 0 if no mapping
clearEvent	The number of the event that clears the bit in the condition register; 0 if no mapping

Details

You can map events to bits in the event registers with this command. This allows you to cause bits in the condition and event registers to be set or cleared when the specified events occur. You can use any valid event number as the event that sets or clears bits.

When a mapped event is programmed to set bits, the corresponding bits in both the condition register and event register are set when the event is detected.

When a mapped event is programmed to clear bits, the bit in the condition register is set to 0 when the event is detected.

If the event is set to zero (0), the bit is never set.

See Event numbers (on page C-10) for information about event numbers.

The query requests the mapped set event and mapped clear event status for a bit in the Operation Event Registers. When you query the mapping for a specific bit, the instrument returns the events that were mapped to set and clear that bit. Zero (0) indicates that the bits have not been set.

Example

`:STATus:OPERation:MAP 0, 4916, 4917`	When event 4916 (the buffer is 0 % filled) occurs, bit 0 is set in the condition register and the event register of the Operation Event Register. When event 4917 (buffer is 100 % filled) occurs, bit 0 in the condition register is cleared.

Also see

Programmable status register sets (on page C-5)

:STATus:PRESet

This command resets all bits in the status model.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:STATus:PRESet
```

Details

This function clears the event registers and the enable registers for operation and questionable. It will not clear the Service Request Enable Register (*SRE) to Standard Request Enable Register (*ESE).

Preset does not affect the event queue.

The Standard Event Status Register is not affected by this command.

Example

`STAT:PRES`	Resets the registers.

Also see

Status model (on page C-1)

:STATus:QUEStionable:CONDition?

This command reads the Questionable Condition Register of the status model.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

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

Details

This command reads the contents of the Questionable Condition Register, which is one of the Questionable Event Registers.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-16).

Example

`:STAT:QUES:COND?`	Reads the Questionable Condition Register.

Also see

Questionable Event Register (on page C-7)
Understanding bit settings (on page C-16)

:STATus:QUEStionable:ENABle

This command sets or reads the contents of the questionable event enable register of the status model.

Type	Affected by	Where saved	Default value
Command and query	STATus:PRESet	Not applicable	0

Usage

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

<n>	The value of the register (0 to 65535)

Details

This command sets or reads the contents of the Enable register of the Questionable Event Register.

When one of these bits is set, when the corresponding bit in the Questionable Event Register or Questionable Condition Register is set, the MSB and QSM bits in the Status Byte Register are set.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-16).

Example

`:STAT:QUES:ENAB 8` `:STAT:QUES:ENAB?`	Enable bit 4, Limit 3 Fail, when the limit test 3 failure value is exceeded. Check to see that the value was set.

Also see

None

:STATus:QUEStionable:MAP

This command queries mapped event numbers or maps event numbers to bits in the event registers.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Not applicable	0

Usage

```
:STATus:QUEStionable:MAP <bitNumber>, <setEvent>
:STATus:QUEStionable:MAP <bitNumber>, <setEvent>, <clearEvent>
:STATus:QUEStionable:MAP? <bitNumber>
```

`<bitNumber>`	The bit number that is mapped to an event (0 to 14)
`<setEvent>`	The number of the event that sets the bits in the condition and event registers; 0 if no mapping
`<clearEvent>`	The number of the event that clears the bit in the condition register; 0 if no mapping

Details

You can map events to bits in the event registers with this command. This allows you to cause bits in the condition and event registers to be set or cleared when the specified events occur. You can use any valid event number as the event that sets or clears bits.

When a mapped event is programmed to set bits, the corresponding bits in both the condition register and event register are set when the event is detected.

When a mapped event is programmed to clear bits, the bit in the condition register is set to 0 when the event is detected.

If the event is set to zero (0), the bit is never set.

See Event numbers (on page C-10) for information about event numbers.

When you query the mapping for a specific bit, the instrument returns the events that were mapped to set and clear that bit. Zero (0) indicates that the bits have not been set.

Example

`:STAT:QUES:MAP 0, 4916, 4917`	When event 4916 (the buffer is 0 % filled) occurs, bit 0 is set in the condition register and the event register of the Questionable Event Register. When event 4917 (buffer is 100 % filled) occurs, bit 0 in the condition register is cleared.

Also see

None

:STATus:QUEStionable[:EVENt]?

This command reads the Questionable Event Register.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

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

Details

This query reads the contents of the questionable status event register. After sending this command and addressing the instrument to talk, a value is sent to the computer. This value indicates which bits in the appropriate register are set.

The Questionable Register can be set to the numeric equivalent of the bit to set. To set more than one bit of the register, set the Questionable Register to the sum of their decimal weights. For example, to set bits B12 and B13, set the Questionable Register to 12,288 (which is the sum of 4,096 + 8,192).

Example

`:STAT:QUES?`	Query the Questionable Register.

Also see

Questionable Event Register (on page C-7)

SYSTem subsystem

This subsystem contains commands that affect the overall operation of the instrument, such as passwords, beepers, communications, event logs, and time.

:SYSTem:ACCess

This command contains the type of access users have to the instrument through different interfaces.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	FULL

Usage

```
:SYSTem:ACCess <permissions>
:SYSTem:ACCess?
```

`<permissions>`	The level of access that is allowed: • Full access for all users from all interfaces: `FULL` • Allows access by one remote interface at a time with login and logout required from other interfaces: `EXCLusive` • Allows access by one remote interface at a time with passwords required on all interfaces: `PROTected` • Allows access by one interface at a time (including the front panel) with passwords required on all interfaces: `LOCKout`

Details

When access is set to full, the instrument accepts commands from any interface with no login or password.

When access is set to exclusive, you must log out of one remote interface and log into another one to change interfaces. You do not need a password with this access.

Protected access is similar to exclusive access, except that you must enter a password when logging in.

When the access is set to locked out, a password is required to change interfaces, including the front-panel interface.

Under any access type, if a script is running on one remote interface when a command comes in from another remote interface, the command is ignored and the message "FAILURE: A script is running, use ABORT to stop it" is generated.

Example

`:SYST:ACC LOCK`
`login admin`
`logout`
	Set the instrument access to locked out.
	Log into the interface using the default password.
	Log out of the interface.

Also see

:SYSTem:PASSword:NEW (on page 6-118)

:SYSTem:BEEPer[:IMMediate]

This command generates an audible tone.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:BEEPer[:IMMediate] <frequency>, <duration>
```

`<frequency>`	The frequency of the beep (20 to 8000 Hz)
`<duration>`	The amount of time to play the tone (0.001 to 100 s)

Details

You can use the beeper of the instrument to provide an audible signal at a specific frequency and time duration. For example, you can use the beeper to signal the end of a lengthy sweep.

Using this function from a remote interface does not affect audible errors or key click settings that were made from the Model 2450 front panel.

Example

`:SYSTem:BEEPer 500, 1`	Beep at 500 Hz for 1 s.

Also see

None

:SYSTem:CLEar

This command clears the event log.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:CLEar
```

Details

This command removes all events from the event log, including entries in the front-panel event log.

Also see

:SYSTem:ERRor[:NEXT]? (on page 6-111)

:SYSTem:COMMunication:LAN:CONFigure

This command specifies the LAN configuration for the instrument.

Type	Affected by	Where saved	Default value
Command and query	Rear panel LAN reset	Nonvolatile memory	AUTO

Usage

```
:SYSTem:COMMunication:LAN:CONFigure "AUTO"
:SYSTem:COMMunication:LAN:CONFigure "MANual,<IPaddress>"
:SYSTem:COMMunication:LAN:CONFigure "MANual,<IPaddress>,<NETmask>"
:SYSTem:COMMunication:LAN:CONFigure "MANual,<IPaddress>,<NETmask>,<GATeway>"
:SYSTem:COMMunication:LAN:CONFigure?
```

AUTO	Use automatically configured LAN settings (default)
MANual	Use manually configured LAN settings
<IPaddress>	LAN IP address; must be a string specifying the IP address in dotted decimal notation; required if the mode is set to manual (default "0.0.0.0")
<NETmask>	The LAN subnet mask; must be a string in dotted decimal notation (default "255.255.255.0")
<GATeway>	The LAN default gateway; must be a string in dotted decimal notation (default "0.0.0.0")

Details

This command specifies how the LAN IP address and other LAN settings are assigned. If automatic configuration is selected, the instrument automatically determines the LAN information. When method is automatic, the instrument first attempts to configure the LAN settings using dynamic host configuration protocol (DHCP). If DHCP fails, it tries dynamic link local addressing (DLLA). If DLLA fails, an error occurs.

If manual is selected, you must define the IP address. You can also assign a subnet mask, and default gateway. The IP address, subnet mask, and default gateway must be formatted in four groups of numbers, each separated by a decimal. If you do not specify a subnet mask or default gateway, the previous settings are used. When specifying multiple parameters, do not use spaces after the commas.

The query form of the command returns the present settings in the order shown here.

Automatic:

```
AUTO,<IPaddress>,<NETmask>,<GATeway>
```

Manual:

```
MANual,<IPaddress>,<NETmask>,<GATeway>
```

Example

```
SYST:COMM:LAN:CONF "MANUAL,192.168.0.1,255.255.240.0,192.168.0.3"
SYST:COMM:LAN:CONF?
```

Set the IP address to be set manually, with the IP address set to 192.168.0.1, the subnet mask to 255.255.240.0, and the gateway address to 192.168.0.3.
Query to verify the settings. The response to the query should be:
`manual,192.168.0.1,255.255.240.0,192.168.0.3`

Also see

:SYSTem:COMMunication:LAN:MACaddress? (on page 6-110)

:SYSTem:COMMunication:LAN:MACaddress?

This command queries the LAN MAC address.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:COMMunication:LAN:MACaddress?
```

Details

The MAC address is a character string representing the MAC address of the instrument in hexadecimal notation. The string includes colons that separate the address octets.

Example

`:SYSTem:COMMunication:LAN:MACaddress?`	Returns the MAC address. For example, you might see: `08:00:11:00:00:57`

Also see

:SYSTem:COMMunication:LAN:CONFigure (on page 6-109)

:SYSTem:ERRor[:NEXT]?

This command returns the oldest unread error message from the event log and removes it from the log.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:ERRor[:NEXT]?
```

Details

As error and status messages occur, they are placed in the event log. The event log is a first-in, first-out (FIFO) register that can hold up to 1000 messages.

This command returns the next entry from the event log.

This command does not affect the event log that is displayed on the front panel.

If there are no entries in the event log, the following message is returned:

```
0,"No error;0,0,0"
```

This command returns only error messages from the event log. To return information and warning messages, see `:SYSTem:EVENtlog:NEXT?`.

Note that if you have used `:SYSTem:ERRor[:NEXT]?` to check events, `:SYSTem:EVENtlog:NEXT?` shows the next event item after the last error that was returned by `:SYSTem:ERRor[:NEXT]?` You will not see warnings or information event log items that occurred before you used `:SYSTem:ERRor[:NEXT]?`

Example

`SYST:ERR:NEXT?`	Returns information on the next error in the event log. For example, if you sent a command without a parameter, the return is: `-109,"Missing parameter;1;2015/05/06 12:57:04.484"`

Also see

:SYSTem:EVENtlog:NEXT? (on page 6-114)

:SYSTem:ERRor:CODE[:NEXT]?

This command reads the oldest error code.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:ERRor:CODE[:NEXT]?
```

Details

This command returns the numeric code of the next error in the event log. The error is cleared from the queue after being read.

This command returns only error messages from the event log. To return information and warning messages, see `:SYSTem:EVENtlog:NEXT?`

Example

`SYST:ERR:CODE?`	Returns the error code of the next error in the event log. For example, if error -222, Parameter data out of range error, occurred, the output is: `-222`

Also see

:SYSTem:EVENtlog:NEXT? (on page 6-114)

:SYSTem:ERRor:COUNt?

This command returns the number of errors in the event log.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:ERRor:COUNt?
```

Details

This command does not return other types of events, such as information messages. To return other types of events, use `:SYSTem:EVENtlog:COUNt?`

This command does not clear the errors from the event log.

Example

`SYST:ERR:COUN?`	If there are five errors in the event log, the output is: 5

Also see

:SYSTem:EVENtlog:COUNt? (on page 6-113)

:SYSTem:EVENtlog:COUNt?

This command returns the number of unread events in the event log.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:EVENtlog:COUNt?
:SYSTem:EVENtlog:COUNt? <eventType>
:SYSTem:EVENtlog:COUNt? <eventType>, <eventType>
```

`<eventType>`	Limits the list of event log entries to specific types; set to: • Returns the number of errors: `ERRor` • Returns the number of warnings: `WARNing` • Returns the number of informational messages: `INFormational` • Returns all events: `ALL`

Details

A count finds the number of unread events in the event log. You can specify the event types to return, or return the count for all events.

This command reports the number of events that have occurred since the command was last sent or since the event log was last cleared.

Example

`:SYST:EVEN:COUN? ERR`	Displays the present number of errors in the instrument event log. If there are three errors in the event log, output is: 3

Also see

:SYSTem:CLEar (on page 6-108)
:SYSTem:EVENtlog:NEXT? (on page 6-114)
:SYSTem:EVENtlog:SAVE (on page 6-116)

:SYSTem:EVENtlog:NEXT?

This command returns the oldest unread event message from the event log.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:EVENtlog:NEXT?
:SYSTem:EVENtlog:NEXT? <eventType>
:SYSTem:EVENtlog:NEXT? <eventType>, <eventType>
:SYSTem:EVENtlog:NEXT? <eventType>, <eventType>, <eventType>
```

`<eventType>`	Limits the event log entries that are returned to specific types; set to: • Returns only the next error: `ERRor` • Returns only the next warning: `WARNing` • Returns only the next informational message: `INFormational` • Returns any event: `ALL`

Details

When an event occurs on the instrument, it is placed in the event log. The `:SYSTem:EVENtlog:NEXT?` command retrieves an unread event from the event log. Once an event is read, it can no longer be accessed remotely. However, it can be viewed on the front panel.

To read multiple commands, execute this command multiple times.

If there are no entries in the event log, the following is returned:

```
0,"No error;0,0,0"
```

If the event type is not defined, an event of any type is returned.

Note that if you have used `:SYSTem:ERRor[:NEXT]?` to check events, `:SYSTem:EVENtlog:NEXT?` shows the next event item after the last error that was returned by `:SYSTem:ERRor[:NEXT]?` You will not see warnings or information event log items that occurred before you used `:SYSTem:ERRor[:NEXT]?`

If the event type is not defined, an event of any type is returned.

The information that is returned is in the order:

```
<eventNumber>, <message>, <eventType>, <timeSeconds>, <timeNanoSeconds>
```

`<eventNumber>`	The event number
`<message>`	A description of the event
`<eventType>`	The type of event: • Error only: 1 • Warning only: 2 • Information only: 4
`<timeSeconds>`	The seconds portion of the time when the event occurred
`<timeNanoSeconds>`	The fractional seconds portion of the time when the event occurred

Example

`SYST:EVEN:NEXT?`	Returns information on the next event in the event log. For example, if you sent a command without a parameter, the return is: `-109,"Missing parameter;1;2015/05/06 12:55:33.648"`

Also see

:SYSTem:CLEar (on page 6-108)
:SYSTem:EVENtlog:SAVE (on page 6-116)

:SYSTem:EVENtlog:POST

This command allows you to post your own text to the event log.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:EVENtlog:POST "<message>"
:SYSTem:EVENtlog:POST "<message>", <eventType>
```

`<message>`	A string that contains the message that will be associated with this event
`<eventType>`	The type of event that is generated; set to: • The error type: `ERRor` • The warning type: `WARNing` • The informational type: `INFormational` (default)

Details

You can use this command to create your own event log entries and assign a severity level to them. This can be useful for debugging and status reporting.

From the front panel, you must set the Log Warnings and Log Information options on to have the custom warning and information events placed into the event log.

Example

`*CLS`
`SYST:EVEN:POST "my error", INF`
`SYST:EVEN:NEXT?`
	Clear the event log.
	Post an error named `my error`.
	Output:
	`1003,"User: my error;4,1400469179,431599191"`

Also see

None

:SYSTem:EVENtlog:SAVE

This command saves the event log to a file on a USB flash drive.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:EVENtlog:SAVE "<filename>"
:SYSTem:EVENtlog:SAVE "<filename>", <eventType>
```

`<filename>`	A string that holds the name of the file to be saved
`<eventType>`	Limits the event log entries that are saved to specific types; set to: • `ERRor`: Saves only error entries • `WARNing`: Saves only warning entries • `INFormational`: Saves only informational entries • `ALL`: Saves all event log entries (default)

Details

This command saves all event log entries to a USB flash drive.

If you do not define an event type, the instrument saves all event log entries.

The extension `.csv` is automatically added to the file name.

Example

`SYST:EVEN:SAVE "/usb1/July_error_log", ERR`	Saves the error events in the event log to a file on the USB flash drive named `July_error_log.csv`.

Also see

:SYSTem:CLEar (on page 6-108)

:SYSTem:GPIB:ADDRess

This command contains the GPIB address.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	18

Usage

```
:SYSTem:GPIB:ADDRess <n>
:SYSTem:GPIB:ADDRess?
```

`<n>`	The GPIB address of the instrument (1 to 30)

Details

The address can be set to any address value from 1 to 30. However, the address must be unique in the system. It cannot conflict with an address that is assigned to another instrument or to the GPIB controller.

A new GPIB address takes effect when the command to change it is processed. If there are response messages in the output queue when this command is processed, they must be read at the new address.

If command messages are being queued (sent before this command has executed), the new settings may take effect in the middle of a subsequent command message, so care should be exercised when setting this attribute from the GPIB interface.

You should allow ample time for the command to be processed before attempting to communicate with the instrument again. After sending this command, make sure to use the new address to communicate with the instrument.

*RST does not affect the GPIB address.

Example

`:SYSTem:GPIB:ADDRess 26` `:SYSTem:GPIB:ADDRess?`	Sets the GPIB address and reads the address. Output: `2.600000e+01`

Also see

GPIB setup (on page 2-56)

:SYSTem:LFRequency?

This query contains the power line frequency setting that is used for NPLC calculations.

Type	Affected by	Where saved	Default value
Query only	Power cycle	Not applicable	Not applicable

Usage

```
:SYSTem:LFRequency?
```

Details

The instrument automatically detects the power line frequency (either 50 Hz or 60 Hz) when the instrument is powered on.

Example

`:SYST:LFR?`	Check the line frequency.

Also see

None

:SYSTem:PASSword:NEW

This command stores the instrument password.

Type	Affected by	Where saved	Default value
Command only	Rear-panel LAN reset	Nonvolatile memory	admin

Usage

```
:SYSTem:PASSword:NEW "<password>"
```

`<password>`	A string that contains the instrument password (maximum 30 characters)

Details

When the access to the instrument is set to protected or lockout, this is the password that is used to gain access.

If you forget the password, you can reset the password to the default:

1. On the front panel, press **MENU**.
2. Under System, select **Info/Manage**.
3. Select **Password Reset**.

You can also reset the password and the LAN settings from the rear panel by inserting a straightened paper clip into hole below LAN RESET.

Example

`SYST:PASS:NEW "N3wpa55w0rd"`	Change the password of the instrument to `N3wpa55w0rd`.

Also see

:SYSTem:ACCess (on page 6-107)

:SYSTem:POSetup

This command selects the defaults that are used when you power on the instrument.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	RST

Usage

```
:SYSTem:POSetup <name>
:SYSTem:POSetup?
```

`<name>`	Which setup to restore when you power on the instrument: • Power on to *RST defaults: RST • Stored setup 0: SAV0 • Stored setup 1: SAV1 • Stored setup 2: SAV2 • Stored setup 3: SAV3 • Stored setup 4: SAV4

Details

When you select RST, the instrument restores settings to their default values when the instrument is powered on.

When you select a SAV option, the settings in the selected saved setup are applied when the instrument is powered on. The settings are saved using the *SAV command.

Example

`SYST:POS SAV1`	Set the instrument to restore the settings that are saved in the stored setup 1 when the instrument is powered on.

Also see

*SAV (on page 6-9)

:SYSTem:TIME

This command sets the absolute time of the instrument.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	See **Details**

Usage

```
:SYSTem:TIME <year>, <month>, <day>, <hour>, <minute>, <second>
:SYSTem:TIME <hour>, <minute>, <second>
:SYSTem:TIME?
:SYSTem:TIME? 1
```

`<year>`	Year; must be more than 1970
`<month>`	Month (1 to 12)
`<day>`	Day (1 to 31)
`<hour>`	Hour in 24-hour time format (0 to 23)
`<minute>`	Minute (0 to 59)
`<second>`	Second (0 to 59)

Details

When queried without a parameter, this command returns the present timestamp value in seconds since January 1, 1970 to the nearest second.

If you query with 1, this command returns the present timestamp in the format:

`<weekday> <month> <day> <hour>:<minute>:<second> <year>`

Where `<weekday>` is the day of the week.

Internally, the instrument bases time in UTC time. UTC time is specified as the number of seconds since Jan 1, 1970, UTC. You can use UTC time from a local time specification, or you can use UTC time from another source (for example, your computer).

Example

`syst:time 2014, 8, 29, 11, 30, 30` `syst:time? 1`	Set the system time to August 29, 2014 at 11:30:30 and confirm setting. Output: `Fri Aug 29 11:30:37 2014`

Also see

None

:SYSTem:VERSion?

Query the present SCPI version.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:SYSTem:VERSion?
```

Details

This query command returns the SCPI version.

Example

`SYSTem:VERSion?`	Query the version. An example of a return is: `1996.0`

Also see

None

TRACe subsystem

The TRACe subsystem contains commands that control the reading buffers.

:TRACe:ACTual?

This command contains the number of readings in the specified reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:ACTual?
:TRACe:ACTual? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This command returns the number of readings stored in the buffer.

Example

`TRACe:MAKE "testData", 200`
`COUN 10`
`MEASure:CURRent? "testData"`
	Creates 200 element reading buffer named `testData`.
	Set the measurement count to 10.
	Set the measurement function to current. Make readings, and store the readings in `testData`. Returns the 10th measurement reading after taking all 10 readings.

`:TRACe:ACTual?`	Returns the number of readings in `defbuffer1`. Example output: `850`
`:TRACe:ACTual? "testData"`	Returns the number of readings in the buffer `testData`. Example output: `10`

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)

:TRACe:ACTual:END?

This command indicates the last index in a reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:ACTual:END?
:TRACe:ACTual:END? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

Use this command to find the ending index in a reading buffer.

Example

```
TRACe:MAKE "test1", 100
COUNt 6
MEASure:CURRent? "test1"
:TRACe:ACTual:STARt? "test1" ; END? "test1"
MEASure:CURRent? "test1"
:TRACe:ACTual:STARt? "test1" ; END? "test1"
```

Create a buffer named `test1` with a capacity of 100 readings.
Set the measure count to 6.
Make measurements and store them in buffer `test1`.
Get the start index and end index of `test1`.
Output: `1;6`
Make six more measurements and store them in buffer `test1`.
Get the start and end index of `test1`.
Output: `1;12`

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:ACTual:STARt? (on page 6-122)
:TRACe:MAKE (on page 6-130)

:TRACe:ACTual:STARt?

This command indicates the starting index in a reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:ACTual:STARt?
:TRACe:ACTual:STARt? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

Use this command to find the starting index in a reading buffer.

Example

```
TRACe:MAKE "test1", 100
COUNt 6
MEASure:CURRent? "test1"
:TRACe:ACTual:STARt? "test1" ; END? "test1"
```

Create a buffer named `test1` with a capacity of 100 readings.
Set the measure count to 6.
Make measurements and store them in buffer `test1`.
Get the start index and end index of `test1`.
`Output: 1;6`

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:ACTual:END? (on page 6-121)
:TRACe:MAKE (on page 6-130)

:TRACe:CLEar

This command clears all readings and statistics from the specified buffer.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRACe:CLEar
:TRACe:CLEar "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:MAKE "testData", 200`
`MEASure:RESistance? "testData"`
`TRACe:ACTual? "testData"`
`TRACe:CLEar "testData"`
`TRACe:ACTual? "testData"`
	Create user-defined buffer named `testData`.
	Make a measurement and store it in `testData` and return the last reading measured.
	Verify that there is data in `testData` buffer.
	Output:
	`1`
	Clear `testData` buffer.
	Verify that `testData` is empty.
	Output:
	`0`

`TRACe:CLEar`
`TRACe:CLEar "defbuffer1"`
`TRACe:CLEar "defbuffer2"`
	Clear the default buffer. This command clears `defbuffer1`.
	Clear `defbuffer1`. Specify default buffer by name.
	Clear `defbuffer2`. Specify default buffer by name.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)

:TRACe:DATA?

This command returns specified data elements from a specified reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:DATA? <startIndex>, <endIndex>
:TRACe:DATA? <startIndex>, <endIndex>, "<bufferName>"
:TRACe:DATA? <startIndex>, <endIndex>, "<bufferName>", <bufferElements>
:TRACe:DATA? <startIndex>, <endIndex>, "<bufferName>",
   <bufferElements>,,<bufferElements>
```

`<startIndex>`	Beginning index of the buffer to return; must be 1 or greater
`<endIndex>`	Ending index of the buffer to return
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<bufferElements>`	A list of elements in the buffer to print; if nothing is specified, `READing` is used; see **Details** for the list of options for buffer elements; a maximum of 14 comma-delimited buffer elements may be specified

Details

The output of `:TRACe:DATA?` is affected by the data format selected by `:FORMat[:DATA]`. If If you set `FORMat[:DATA]` to `REAL` or `SREAL`, you will have fewer options for buffer elements. The only buffer elements available are `READing` and `SOURce`. If you request a buffer element that is not permitted for the selected data format, the instrument generates the error 1133, "Parameter 4, Syntax error, expected valid name parameters."

NOTE

To change the number of digits returned in a remote command reading, use the `:FORMat:ASCii:PRECision` command.

When specifying buffer elements, you can:

- Specify buffer elements in any order.
- Include any or all of the buffer elements listed below in a single list. You can repeat elements as long as the number of elements in the list is less than 14.
- Use a comma to delineate multiple elements for a data point.

The options for <bufferElements> are described in the following table.

Option	Description
DATE	The date when the data point was measured
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds for the data point when the data point was measured
READing	The measurement reading based on the SENS:FUNC setting; if no buffer elements are defined, this option is used
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
SOURce	The source value; if readback is ON, then it is the readback value, otherwise it is the programmed source value (see :SOURce[1]:<function>:READ:BACK (on page 6-87))
SOURFORMatted	The source value as it appears on the display
SOURSTATus	The status information associated with sourcing. The values returned indicate the status of the following conditions: • Overvoltage protection was active • Measured source value was read • Overtemperature condition existed • Source function level was limited • Four-wire sense was used • Output was on
SOURUNIT	The unit of value associated with the source value
STATus	The status information associated with the measurement; see the "Buffer status bits for sense measurements" table below
TIME	The time for the data point
TSTamp	The timestamp for the data point
UNIT	The unit of measure associated with the measurement

The STATus buffer element returns status values for the readings in the buffer. The status values are floating-point numbers that encode the status value. Refer to the following table for values.

Buffer status bits for sense measurements

Bit (hex)	Name	Decimal	Description
0x0001	STAT_QUESTIONABLE	1	Measure status questionable
0x0006	STAT_ORIGIN	6	A/D converter from which reading originated; for the Model 2450, this will always be 0 (Main)
0x0008	STAT_TERMINAL	8	Measure terminal, front is 1, rear is 0
0x0010	STAT_LIMIT2_LOW	16	Measure status limit 2 low
0x0020	STAT_LIMIT2_HIGH	32	Measure status limit 2 high
0x0040	STAT_LIMIT1_LOW	64	Measure status limit 1 low
0x0080	STAT_LIMIT1_HIGH	128	Measure status limit 1 high
0x0100	STAT_START_GROUP	256	First reading in a group

Example

```
TRAC:MAKE "buf100", 100
TRIGger:LOAD "SimpleLoop", 5, 0, "buf100"
SOUR:VOLT 0.35
INIT
*WAI
TRAC:DATA? 1, 5, "buf100", READ, SOUR, REL
TRAC:DATA? 1, 5, "buf100", READ, REL
TRAC:DATA? 1, 5, "buf100", REL
TRAC:DATA? 1, 3, "buf100"
```

Create a buffer called buf100 with a maximum size of 100.
Set the instrument to configure the trigger model to loop, taking 5 readings with no delay, and store the readings in the buf100 reading buffer.
Set the source level for voltage to 0.35.
Initiate the trigger model and wait for the trigger model to complete. The trigger model will make 5 readings and store them in buf100.
Read the 5 data points, reading, programmed source, and relative time for each point.
Output:

```
-0.000000,0.350000,0.000000;
    -0.000000,0.350000,0.266978,
    -0.000000,0.350000,0.443087,
    -0.000000,0.350000,0.704459,
    -0.000000,0.350000,0.881419
```

Read 5 data points and include the reading and relative time for each data point.
Output:

```
-0.000000,0.000000,
    -0.000000,0.266978,
    -0.000000,0.443087,
    -0.000000,0.704459,
    -0.000000,0.881419
```

Read 5 data points and include relative time for each data point.
Output:

```
0.000000,0.266978;0.443087,
    0.704459,0.881419
```

Returns the first 3 reading values from buf100 reading buffer
Output:

```
-0.000000,-0.000000,-0.000000
```

Also see

:FORMat[:DATA] (on page 6-38)
Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)

:TRACe:DELete

This command deletes a user-defined reading buffer.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRACe:DELete "<bufferName>"
```

`<bufferName>`	A string that contains the name of the user-defined reading buffer to delete

Details

You cannot delete the default reading buffers, `defbuffer1` and `defbuffer2`.

Example

`TRAC:DEL "testData"`	Delete the `testData` buffer.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)

:TRACe:FILL:MODE

This command determines if a reading buffer is filled continuously or is filled once and stops.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	defbuffer1: CONT defbuffer2: CONT User-defined buffers: ONCE

Usage

```
:TRACe:FILL:MODE <fillType>
:TRACe:FILL:MODE <fillType>, "<bufferName>"
:TRACe:FILL:MODE?
:TRACe:FILL:MODE? "<bufferName>"
```

`<fillType>`	Fill the buffer continuously: `CONTinuous` Fill the buffer, then stop: `ONCE`
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

When a reading buffer is set to fill once, no data is overwritten in the buffer. When the buffer is filled, no more data is stored in that buffer and new readings are discarded.

When a reading buffer is set to fill continuously, the oldest data is overwritten by the newest data after the buffer fills.

When you change the fill mode of a buffer, any data in the buffer is cleared.

Example

`TRACe:MAKE "testData", 100`	Create a user-defined reading buffer named `testData` with a capacity of 100 measurements.
`TRACe:FILL:MODE? "testData"`	Query the fill mode setting for `testData`.
`TRACe:FILL:MODE CONT, "testData"`	Output: `ONCE`
`TRACe:FILL:MODE? "testData"`	Set `testData` fill mode to continuous.
	Query the fill mode setting for `testData`.
`TRACe:FILL:MODE?`	Output: `CONT`
	Query the fill mode setting for `defbuffer1`.
	Output: `CONT`

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:CLEar (on page 6-123)

:TRACe:LOG:STATe

This command indicates if information events are logged when the specified reading buffer is at 0 % or 100 % filled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	defbuffer1: 1 (ON) defbuffer2: 1 (ON) User-created buffer: 0 (OFF)

Usage

```
:TRACe:LOG:STATe <logState>
:TRACe:LOG:STATe <logState>, "<bufferName>"
:TRACe:LOG:STATe?
:TRACe:LOG:STATe? "<bufferName>"
```

`<logState>`	Do not log information events: `OFF` or `0` Log information events: `ON` or `1`
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

If this is set to on, when the reading buffer is cleared (0 % filled) or full (100 % filled), an event is logged in the event log. If this is set to off, reading buffer status is not reported in the event log.

Example

`TRACe:LOG:STATe?`	Query the log state of `defbuffer1`. Output: `1` Indicates that the log state is on.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
Using the event log (on page 2-131)

:TRACe:MAKE

This command creates a user-defined reading buffer.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRACe:MAKE "<bufferName>", <bufferSize>
:TRACe:MAKE "<bufferName>", <bufferSize>, <bufferStyle>
```

`<bufferName>`	A user-supplied string that indicates the name of the buffer
`<bufferSize>`	A number that indicates the maximum number of readings that can be stored in `<bufferName>`; minimum is 10
`<bufferStyle>`	The type of reading buffer to create: • Store readings with reduced accuracy (6.5 digits) with no formatting information, 1 µs accurate timestamp, maximum 27,500,000 readings: `COMPact` • Store readings with full accuracy with formatting, maximum 6,875,000 readings: `STANdard` (default) • Store the same information as standard, plus additional information: `FULL` • Store external reading buffer data: `WRITable` • Store external reading buffer data with two reading values: `FULLWRITable`

Details

You cannot assign user-defined reading buffers the name `defbuffer1` or `defbuffer2`.

If you create a reading buffer that has the same name as an existing user-defined buffer, an event message 1115, "Parameter error: TRACe:MAKE cannot take an existing reading buffer name" is generated.

When you create a reading buffer, it becomes the active buffer. If you create two reading buffers, the last one you create becomes the active buffer.

The default fill mode of a user-defined buffer is once. You can change it to continuous.

Once the buffer style is selected, it cannot be changed.

Once you store the first reading in a compact buffer, you cannot change certain measurement settings, including range, display digits, and units; you must clear the buffer first.

Not all remote commands are compatible with the compact, writable, and full writable buffer styles. Check the Details section of the command descriptions before using them with any of these buffer styles.

Writable readings are used to bring external data into the instrument. You cannot assign them to collect data from the instrument.

You can change the buffer capacity for an existing buffer through the front panel or by using the `:TRACe:POINts` command.

Example 1

`TRACe:MAKE "capTrace", 200, WRITable`	Create a 200-element writable reading buffer named `capTrace`.

Example 2

`TRACe:MAKE "bufferVolts", 100`
`TRACe:POINts? "bufferVolts"`
 	
`TRACe:DELete "bufferVolts"`
`TRACe:MAKE "bufferVolts", 1000`
`TRACe:POINts?`
	Create a buffer named `bufferVolts` to store 100 readings.
	Query the size of `bufferVolts`.
	Output: `100`
	Delete the buffer named `bufferVolts`.
	Make a new buffer named `bufferVolts` to store 1000 readings.
	Query the size of `bufferVolts` again to verify it can store 1000 readings.
	Output: `1000`

Example 3

`TRACe:POINts 5000, "bufferVolts"` `TRACe:POINts?`	Resize an existing buffer named `bufferVolts` to store 5000 readings. Query the size of `bufferVolts` to verify it can store 5000 readings. Output: `5000`

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:FILL:MODE (on page 6-127)
:TRACe:POINts (on page 6-132)
:TRACe:WRITe:FORMat (on page 6-142)
:TRACe:WRITe:READing (on page 6-144)

:TRACe:POINts

This command contains the number of readings a buffer can store.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRACe:POINts <newSize>
:TRACe:POINts <newSize>, "<bufferName>"
:TRACe:POINts?
:TRACe:POINts? "<bufferName>"
```

`<newSize>`	The new size for the buffer
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This command allows you to change or view how many readings a buffer can store. Changing the size of a buffer will cause any existing data in the buffer to be lost.

The overall capacity of all buffers stored in the instrument cannot exceed 6,875,000 readings for standard reading buffers and 27,500,000 for compact reading buffers. For more information about buffer capacity, see Setting reading buffer capacity (on page 3-16).

Example

`TRACe:MAKE "testData", 100`
`TRACe:POINts 300, "testData"`
`TRACe:POINts? "testData"`
 	
`TRACe:POINts?`
	Create a user-defined reading buffer named `testData` with a capacity of 100 measurements.
	Change the buffer capacity to 300.
	Query the capacity of `testData`.
	Output:
	`300`
	Query the capacity of the default buffer.
	Output:
	`10000`

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)

:TRACe:SAVE

This command saves data from the specified reading buffer to a USB flash drive.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRACe:SAVE "<fileName>"
:TRACe:SAVE "<fileName>", "<bufferName>"
:TRACe:SAVE "<fileName>", "<bufferName>", <timeFormat>
:TRACe:SAVE "<fileName>", "<bufferName>", <timeFormat>, <start>, <end>
```

`<fileName>`	A string that indicates the name of the file on the USB flash drive in which to save the reading buffer
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<timeFormat>`	Defines how date and time information from the buffer is saved in the file on the USB flash drive; the values are: • Dates, times, and fractional seconds are saved; the default value: `FORMat` • Relative timestamps are saved: `RELative` • Seconds and fractional seconds are saved: `RAW` • Timestamps are saved: `STAMp`
`<start>`	Defines the starting point in the buffer to start saving data
`<end>`	Defines the ending point in the buffer to stop saving data

Details

The filename must specify the full path (including `/usb1/`). If included, the file extension must be set to `.csv` (if no file extension is specified, `.csv` is added).

For options that save more than one item of time information, each item is comma-delimited. For example, the default format is `date`, `time`, and `fractional seconds` for each reading.

The Model 2450 does not check for existing files when you save. Verify that you are using a unique name to avoid overwriting any existing `.csv` files on the flash drive.

Example

`TRACe:MAKE "MyBuffer", 100`
`SENSe:COUNt 5`
`MEASure:CURRent:DC? "MyBuffer"`
`TRACe:DATA? 1,5, "MyBuffer", READ, REL, SOUR`
`TRACe:SAVE "/usb1/myData.csv", "MyBuffer"`
`TRACe:SAVE "/usb1/myDataRel.csv", "MyBuffer", REL`
	Create a buffer called `MyBuffer` with a maximum size of 100.
	Make five readings for each measurement request and return the data.
	Make the measurements.
	Read the reading, relative timestamp, and source value for each point from 1 to 5.
	Output:
	`-0.000000,0.000000,`
	`0.000000,-0.000000,`
	`0.301759,0.000000,`
	`-0.000000,0.579068,`
	`0.000000,-0.000000,`
	`0.884302,0.000000,`
	`-0.000000,1.157444,`
	`0.000000`
	Save all reading and default time information from a buffer named `MyBuffer` to a file named `myData.csv` on the USB flash drive.
	Save all readings and relative timestamps from `MyBuffer` to a file named `myDataRel.csv` on the USB flash drive.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:SAVE:APPend (on page 6-135)

:TRACe:SAVE:APPend

This command appends data from the reading buffer to a file on the USB flash drive.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRACe:SAVE:APPend "<fileName>"
:TRACe:SAVE:APPend "<fileName>", "<bufferName>"
:TRACe:SAVE:APPend "<fileName>", "<bufferName>", <timeFormat>
:TRACe:SAVE:APPend "<fileName>", "<bufferName>", <timeFormat>, <start>, <end>
```

`<fileName>`	A string that indicates the name of the file on the USB flash drive in which to save the reading buffer
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<timeFormat>`	Indicates how date and time information from the buffer is saved in the file on the USB flash drive; the values are: • Dates, times, and fractional seconds are saved; the default value: `FORMat` • Relative timestamps are saved: `RELative` • Seconds and fractional seconds are saved: `RAW` • Timestamps are saved: `STAMp`
`<start>`	Defines the starting point in the buffer to start saving data
`<end>`	Defines the ending point in the buffer to stop saving data

Details

If the file you specify does not exist on the USB flash drive, this command creates the file.

For options that save more than one item of time information, each item is comma-delimited. For example, the default format is date, time, and fractional seconds for each reading.

The file extension `.csv` is appended to the filename if necessary. Any file extension other than `.csv` generates an error.

The index column entry in the `.csv` file starts at 1 for each append operation.

Example

```
TRACe:MAKE "testData", 100
SENSe:COUNt 5
MEASure:CURRent:DC? "testData", READ, REL, SOUR
TRACe:SAVE "/usb1/myData5.csv", "testData"
TRACe:CLEAr
MEASure:CURRent:DC?
TRACe:SAVE:APPend "/usb1/myData5.csv", "defbuffer1"
MEASure:CURRent:DC? "testData"
TRACe:SAVE:APPend "/usb1/myData5.csv", "testData", RAW, 6, 10
```

Create a buffer called `testData`.
Make 5 readings and return the fifth point, which will contain the reading, relative timestamp, and source value. Store the buffer data in the `myData5.csv` file.
Clear `defbuffer1`.
Make 5 readings, store them in `defbuffer1`, and return the fifth reading.
Append all the readings stored in `defbuffer1` to the `myData5.csv` file.
Take 5 more readings, store them in `testData`, and return the fifth reading.
Append all the readings stored in positions 6 through 10 `testData` to the `myData5.csv` file using raw timestamps.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)

:TRACe:STATistics:AVERage?

This command returns the average of all readings in the buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:STATistics:AVERage?
:TRACe:STATistics:AVERage? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This command returns the average reading calculated from all of the readings in the specified reading buffer.

When the reading buffer is configured to fill continuously and overwrite older data with new data, the buffer statistics include the data that was overwritten. To get statistics that do not include data that has been overwritten, define a large buffer size that will accommodate the number of readings you will make.

Example

`TRACe:STAT:AVERage?`	Returns the average reading for the readings in the default buffer `defbuffer1`.
`TRACe:STAT:AVERage? "testData"`	Returns the average reading for the readings in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:STATistics:CLEar (on page 6-137)
:TRACe:STATistics:MAXimum? (on page 6-138)
:TRACe:STATistics:MINimum? (on page 6-139)
:TRACe:STATistics:PK2Pk? (on page 6-140)
:TRACe:STATistics:STDDev? (on page 6-140)

:TRACe:STATistics:CLEar

This command clears the statistical information associated with the specified buffer.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRACe:STATistics:CLEar
:TRACe:STATistics:CLEar "<bufferName>"
```

`<bufferName>`	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; if no buffer is defined, clears the statistics from `defbuffer1`

Details

This command clears the statistics without clearing the readings.

Example

`TRACe:STATistics:CLEar`	Clear all statistics in `defbuffer1`.
`TRACe:STATistics:CLEar "testData"`	Clears all statistics in a user-defined buffer named `testData`.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:STATistics:AVERage? (on page 6-136)
:TRACe:STATistics:MAXimum? (on page 6-138)
:TRACe:STATistics:MINimum? (on page 6-139)
:TRACe:STATistics:PK2Pk? (on page 6-140)
:TRACe:STATistics:STDDev? (on page 6-140)

:TRACe:STATistics:MAXimum?

This command returns the maximum reading value in the reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:STATistics:MAXimum?
:TRACe:STATistics:MAXimum? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:STAT:MAXimum?`	Returns the maximum reading value in the default buffer, `defbuffer1`.
`TRACe:STAT:MAXimum? "testData"`	Returns the maximum reading value in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:STATistics:AVERage? (on page 6-136)
:TRACe:STATistics:CLEar (on page 6-137)
:TRACe:STATistics:MINimum? (on page 6-139)
:TRACe:STATistics:PK2Pk? (on page 6-140)
:TRACe:STATistics:STDDev? (on page 6-140)

:TRACe:STATistics:MINimum?

This command returns the minimum reading value in the reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:STATistics:MINimum?
:TRACe:STATistics:MINimum? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:STAT:MINimum?`	Returns the minimum reading value in the default buffer `defbuffer1`.
`TRACe:STAT:MINimum? "testData"`	Returns the minimum reading value in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:STATistics:AVERage? (on page 6-136)
:TRACe:STATistics:CLEar (on page 6-137)
:TRACe:STATistics:MAXimum? (on page 6-138)
:TRACe:STATistics:PK2Pk? (on page 6-140)
:TRACe:STATistics:STDDev? (on page 6-140)

:TRACe:STATistics:PK2Pk?

This command returns the peak-to-peak value of all readings in the reading buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:STATistics:PK2Pk?
:TRACe:STATistics:PK2Pk? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:STAT:PK2Pk?`	Returns the peak-to-peak reading value in the default buffer `defbuffer1`.
`TRACe:STAT:PK2Pk? "testData"`	Returns the peak-to-peak reading value in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:STATistics:AVERage? (on page 6-136)
:TRACe:STATistics:CLEar (on page 6-137)
:TRACe:STATistics:MAXimum? (on page 6-138)
:TRACe:STATistics:MINimum? (on page 6-139)
:TRACe:STATistics:STDDev? (on page 6-140)

:TRACe:STATistics:STDDev?

This command returns the standard deviation of all readings in the buffer.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
:TRACe:STATistics:STDDev?
:TRACe:STATistics:STDDev? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:STAT:STDDev?`	Returns the standard deviation of the readings in the default buffer `defbuffer1`.
`TRACe:STAT:STDDev? "testData"`	Returns the standard deviation of the readings in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-10)
Remote buffer operation (on page 3-31)
:TRACe:MAKE (on page 6-130)
:TRACe:STATistics:CLEar (on page 6-137)
:TRACe:STATistics:MAXimum? (on page 6-138)
:TRACe:STATistics:MINimum? (on page 6-139)
:TRACe:STATistics:PK2Pk? (on page 6-140)

:TRACe:TRIGger

This command makes readings using the active measure function and stores them in a reading buffer.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRACe:TRIGger
:TRACe:TRIGger "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

A measure function must be selected before sending this command.

This command makes the number of measurements that is set by the count command.

Example

`TRACe:MAKE "MyBuffer", 100`
`COUN 5`
`TRACe:TRIG "MyBuffer"`
`TRACe:DATA? 1,5, "MyBuffer", rel`
	Create a buffer called `MyBuffer` with a maximum size of 100.
	Make readings and store them in `MyBuffer`.
	Recall the relative time when the data points were measured for the first five readings in the buffer.
	Example output:
	`0.000000,0.408402,0.816757,1.208823,1.617529`

Also see

[:SENSe[1]]:COUNt (on page 6-69)
[:SENSe[1]]:FUNCtion[:ON] (on page 6-70)
:TRACe:MAKE (on page 6-130)

:TRACe:WRITe:FORMat

This command sets the units and number of digits of the readings that are written into the reading buffer.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRACe:WRITe:FORMat "<bufferName>", <units>, <displayDigits>
:TRACe:WRITe:FORMat "<bufferName>", <units>, <displayDigits>, <extraUnits>
:TRACe:WRITe:FORMat "<bufferName>", <units>, <displayDigits>, <extraUnits>,
   <extraDigits>
```

`<bufferName>`	A user-supplied string that indicates the name of the buffer	
`<units>`	The units for the first measurement in the buffer index: • `AMP` • `AMP_AC` • `CELSius` • `DECibel` • `FAHRenheit` • `FARad` • `HERTz` • `KELVin` • `NONE`	• `OHM` • `PERCent` • `RATio` • `RECiprocal` • `SECond` • `VOLT` • `VOLT_AC` • `WATT` • `X`
`<displayDigits>`	Integer from 3 to 8	
`<extraUnits>`	The units for the second measurement in the buffer index; the selections are the same as `<units>` (only valid for buffer style `FULLWRITable`); if this parameter is not specified, the value for `<units>` is used	
`<extraDigits>`	The number of digits to use for the second measurement; the selections are the same as `<displayDigits>` (only valid for buffer style `FULLWRITable`); if this parameter is not specified, the value for `<displayDigits>` is used	

Details

This command is valid when the buffer style is writable or full writable.

Defines the units and the number of digits that are reported for the data. This function affects how the data is shown in the reading buffer and what is shown on the front-panel Home, Histogram, Reading Table, and Graph screens.

Example 1

```
:TRAC:MAKE "write2me", 1000, WRITable
:TRAC:WRIT:FORM "write2me", WATT, 4
:TRAC:WRIT:READ "write2me", 1
:TRAC:WRIT:READ "write2me", 2
:TRAC:WRIT:READ "write2me", 3
:TRAC:WRIT:READ "write2me", 4
:TRAC:WRIT:READ "write2me", 5
:TRAC:WRIT:READ "write2me", 6
:TRAC:DATA? 1, 6, "write2me", read, unit
```

Creates a 1000-point reading buffer named `write2me`. Style is writable.
Set the data format to show units of watts with 4-½ digit resolution.
Write 6 pieces of data into the buffer.
Read the buffer.
Output:

```
1.000000E+00,Watt DC,2.000000E+00,Watt DC,3.000000E+00,Watt DC,4.000000E+00,Watt
    DC,5.000000E+00,Watt DC,6.000000E+00,Watt DC
```

Example 2

```
:TRAC:MAKE "write2me", 1000, FULLWRIT
:TRAC:WRIT:FORM "write2me", WATT, 4, WATT, 4
:TRAC:WRIT:READ "write2me", 1, 7
:TRAC:WRIT:READ "write2me", 2, 8
:TRAC:WRIT:READ "write2me", 3, 9
:TRAC:WRIT:READ "write2me", 4, 10
:TRAC:WRIT:READ "write2me", 5, 11
:TRAC:WRIT:READ "write2me", 6, 12
:TRAC:DATA? 1, 6, "write2me", read, unit, read, unit
```

Creates a 1000-point reading buffer named `write2me`. Style is full writable.
Set the data format to show units of watts with 4-½ digit resolution for the first value and the second value in the buffer index.
Write 12 pieces of data into the buffer.
Read the buffer.
Output:

```
1.000000E+00,Watt DC,7.000000E+00,Watt DC,2.000000E+00,Watt DC,8.000000E+00,Watt
    DC,3.000000E+00,Watt DC,9.000000E+00,Watt DC,4.000000E+00,Watt
    DC,10.000000E+00,Watt DC,5.000000E+00,Watt DC,11.000000E+00,Watt
    DC,6.000000E+00,Watt DC,12.000000E+00,Watt DC,
```

Also see

Reading buffers (on page 3-10)
:TRACe:MAKE (on page 6-130)
:TRACe:WRITe:READing (on page 6-144)
Writable reading buffers (on page 3-36)

:TRACe:WRITe:READing

This command allows you to write readings into the reading buffer.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

For buffers with the writable buffer style:

```
:TRACe:WRITe:READing "<bufferName>", <readingValue>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <seconds>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <seconds>, <fractionalSeconds>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <seconds>,
   <fractionalSeconds>, <status>
```

For buffers with the full writable buffer style:

```
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <extraValue>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <extraValue>, <seconds>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <extraValue>, <seconds>,
   <fractionalSeconds>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <extraValue>, <seconds>,
   <fractionalSeconds>, <status>
```

`<bufferName>`	A user-supplied string that indicates the name of the buffer
`<readingValue>`	The first value that is recorded in the buffer index
`<extraValue>`	A second value that is recorded in the buffer index (only valid for buffer style `FULLWRITable`)
`<seconds>`	An integer that represents the seconds
`<fractionalSeconds>`	The portion of time that represents the fractional seconds
`<status>`	The reading that is the start of a group of readings: `buffer.STAT_START_GROUP`; set to 256 to graph a family of curves (default is 0)

Details

This command writes the data you specify into a reading buffer. The reading buffer must be set to the writable or full writable style, which is set when you make the buffer.

Data must be added in chronological order. If the time is not specified for a reading, it is set to one integer second after the last reading. As you write the data, the front-panel Home screen updates and displays the reading you entered.

Example 1

```
:TRAC:MAKE "write2me", 1000, WRITable
:TRAC:WRIT:FORM "write2me", WATT, 4
:TRAC:WRIT:READ "write2me", 1
:TRAC:WRIT:READ "write2me", 2
:TRAC:WRIT:READ "write2me", 3
:TRAC:WRIT:READ "write2me", 4
:TRAC:WRIT:READ "write2me", 5
:TRAC:WRIT:READ "write2me", 6
:TRAC:DATA? 1, 6, "write2me", read, unit
```

Creates a 1000-point reading buffer named `write2me`. Style is writable.
Set the data format to show a unit of watts with 4-½ digit resolution.
Write 6 pieces of data into the buffer.
Read the buffer.
Output:

```
1.000000E+00,Watt DC,2.000000E+00,Watt DC,3.000000E+00,Watt DC,4.000000E+00,Watt
    DC,5.000000E+00,Watt DC,6.000000E+00,Watt DC
```

Example 2

```
:TRAC:MAKE "write2me", 1000, FULLWRIT
:TRAC:WRIT:FORM "write2me", WATT, 4, WATT, 4
:TRAC:WRIT:READ "write2me", 1, 7
:TRAC:WRIT:READ "write2me", 2, 8
:TRAC:WRIT:READ "write2me", 3, 9
:TRAC:WRIT:READ "write2me", 4, 10
:TRAC:WRIT:READ "write2me", 5, 11
:TRAC:WRIT:READ "write2me", 6, 12
:TRAC:DATA? 1, 6, "write2me", read, unit, read, unit
```

Creates a 1000-point reading buffer named `write2me`. Style is full writable.
Set the data format to show units of watts with 4-½ digit resolution for the first value and the second value in the buffer index.
Write 12 pieces of data into the buffer.
Read the buffer.
Output:

```
1.000000E+00,Watt DC,7.000000E+00,Watt DC,2.000000E+00,Watt DC,8.000000E+00,Watt
    DC,3.000000E+00,Watt DC,9.000000E+00,Watt DC,4.000000E+00,Watt
    DC,10.000000E+00,Watt DC,5.000000E+00,Watt DC,11.000000E+00,Watt
    DC,6.000000E+00,Watt DC,12.000000E+00,Watt DC,
```

Also see

Reading buffers (on page 3-10)
:TRACe:DATA? (on page 6-124)
:TRACe:MAKE (on page 6-130)
:TRACe:WRITe:FORMat (on page 6-142)
Writable reading buffers (on page 3-36)

TRIGger subsystem

The commands in this subsystem configure and control the trigger operations, including the trigger model.

:ABORt

This command stops all trigger model commands on the instrument.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:ABORt
```

Details

When this command is received, the instrument stops the trigger model.

Also see

Aborting the trigger model (on page 3-112)
Trigger model (on page 3-92)

:INITiate[:IMMediate]

This command starts the trigger model.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:INITiate[:IMMediate]
```

Also see

Trigger model (on page 3-92)

:TRIGger:BLENder<n>:CLEar

This command clears the blender event detector and resets the overrun indicator of blender <n>.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:BLENder<n>:CLEar
```

<n>	The blender number (1 or 2)

Details

This command sets the blender event detector to the undetected state and resets the overrun indicator of the event detector.

Example

`:TRIG:BLEN2:CLE`	Clears the event detector for blender 2.

Also see

None

:TRIGger:BLENder<n>:MODE

This command selects whether the blender performs OR operations or AND operations.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Trigger blender clear	Save settings	AND

Usage

```
:TRIGger:BLENder<n>:MODE <operation>
:TRIGger:BLENder<n>:MODE?
```

`<n>`	The blender number (1 or 2)
`<operation>`	The type of operation: • `OR` • `AND`

Details

This command selects whether the blender waits for any one event (`OR`) or waits for all selected events (`AND`) before signaling an output event.

Example 1

```
:DIG:LINE3:MODE TRIG, IN
:DIG:LINE5:MODE TRIG, IN
:TRIG:BLEN1:MODE OR
:TRIG:BLEN1:STIM1 DIG3
:TRIG:BLEN1:STIM2 DIG5
```

Set digital I/O lines 3 and 5 as trigger in lines. Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.

Also see

:TRIGger:BLENder<n>:STIMulus<m> (on page 6-148)

:TRIGger:BLENder<n>:OVERrun?

This command indicates whether or not an event was ignored because of the event detector state.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:BLENder<n>:OVERrun?
```

<n>	The blender number (1 or 2)

Details

Indicates if an event was ignored because the event detector was already in the detected state when the event occurred. This is an indication of the state of the event detector that is built into the event blender itself.

This command does not indicate if an overrun occurred in any other part of the trigger model or in any other trigger object that is monitoring the event. It also is not an indication of an action overrun.

Example

`:TRIG:BLEN1:OVER?`	If an event was ignored, the output is `1`. If an event was not ignored, the output is `0`.

Also see

:TRIGger:BLENder<n>:CLEar (on page 6-146)

:TRIGger:BLENder<n>:STIMulus<m>

This command specifies the events that trigger the blender.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Trigger blender clear	Save settings	NONE

Usage

```
:TRIGger:BLENder<n>:STIMulus<m> <event>
:TRIGger:BLENder<n>:STIMulus<m>?
```

<n>	The blender number (1 or 2)
<m>	The stimulus input number (1 to 4)
<event>	See **Details**

Details

There are four stimulus inputs that can each select a different event.

Use zero to disable the blender input.

The `<event>` parameter may be any of the trigger events shown in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	`LAN<n>`
Trigger event blender <n> (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer <n> (1 to 4) expired	`TIMer<n>`
Source limit condition occurs	`SLIMit`

Example

`:DIG:LINE3:MODE TRIG, IN`
`:DIG:LINE5:MODE TRIG, IN`
`:TRIG:BLEN1:MODE OR`
`:TRIG:BLEN1:STIM1 DIG3`
`:TRIG:BLEN1:STIM2 DIG5`
	Set digital I/O lines 3 and 5 as trigger in lines. Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.

Also see

:TRIGger:BLENder<n>:MODE (on page 6-147)

:TRIGger:BLOCk:BRANch:ALWays

This command defines a trigger model block that always goes to a specific block.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:ALWays <blockNumber>, <branchToBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<branchToBlock>`	The block number of the trigger model block to execute when the trigger model reaches this block

Details

When the trigger model reaches a branch-always building block, it goes to the building block set by `<branchToBlock>`.

Example

`TRIG:BLOC:BRAN:ALW 9, 20`	When the trigger model reaches block 9, it will always branch to block 20.

Also see

None

:TRIGger:BLOCk:BRANch:COUNter

This command defines a trigger model block that branches to a specified block a specified number of times.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:COUNter <blockNumber>, <targetCount>, <branchToBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<targetCount>`	The number of times to repeat
`<branchToBlock>`	The block number of the trigger model block to execute when the counter is less than to the `<targetCount>` value

Details

This command defines a trigger model building block that branches to another block using a counter to iterate a specified number of times.

Counters increment every time the trigger model reaches them until they are more than or equal to the count value. At that point, the trigger model continues to the next building block in the sequence.

If you are using remote commands, you can query the counter. The counter is incremented immediately before the branch compares the actual counter value to the set counter value. Therefore, the counter is at 0 until the first comparison. When the trigger model reaches the set counter value, branching stops and the counter value is one greater than the setting. Use `:TRIGger:BLOCk:BRANch:COUNter:COUNt?` to query the counter.

Example

`TRIG:LOAD "EMPTY"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 readings.
	Delay a second.
	Loop three more times back to block 2.
	At end of execution, 15 readings are stored in `defbuffer1`.

Also see

:TRIGger:BLOCk:BRANch:COUNter:COUNt? (on page 6-151)

:TRIGger:BLOCk:BRANch:COUNter:COUNt?

This command returns the count that the trigger model is on.

Type	Affected by	Where saved	Default value
Query only	Recall settings Instrument reset Power cycle	Not applicable	Not applicable

Usage

`:TRIGger:BLOCk:BRANch:COUNter:COUNt? <blockNumber>`

`<blockNumber>`	The sequence of the block in the trigger model

Details

The query returns the number of times the trigger model has looped. The counter is defined by `:TRIGger:BLOCk:BRANch:COUNter`.

Example

`TRIG:LOAD "Empty"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
`INIT`
`TRIG:BLOCK:BRAN:COUN:COUN?`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 readings.
	Delay 1 s.
	Loop three more times back to block 2.
	At end of execution, 15 readings are stored in `defbuffer1`.
	Check to see which count the trigger model has completed.

Also see

:TRIGger:BLOCk:BRANch:COUNter (on page 6-150)

:TRIGger:BLOCk:BRANch:COUNter:RESet

This command creates a block in the trigger model that resets a branch counter to 0.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

`:TRIGger:BLOCk:BRANch:COUNter:RESet <blockNumber>, <counter>`

`<blockNumber>`	The sequence of the block in the trigger model
`<counter>`	The block number of the counter that is to be reset

Details

When the trigger model reaches the Counter Reset block, it resets the count of the specified Branch on Counter block to zero.

Example

`TRIG:LOAD "EMPTY"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
`TRIG:BLOC:BRAN:COUN:RES 6, 3`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 readings.
	Delay a second.
	Loop three more times back to block 2.
	Reset block 3 to 0.

Also see

:TRIGger:BLOCk:BRANch:COUNter (on page 6-150)
:TRIGger:BLOCk:BRANch:COUNter:COUNt? (on page 6-151)

:TRIGger:BLOCk:BRANch:DELTa

This command defines a trigger model block that goes to a specified block if the difference of two measurements meets preset criteria.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:DELTa <blockNumber>, <targetDifference>, <branchToBlock>
:TRIGger:BLOCk:BRANch:DELTa <blockNumber>, <targetDifference>, <branchToBlock>,
   <measureBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<targetDifference>`	The value against which the block compares the difference between the measurements
`<branchToBlock>`	The block number of the trigger model block to execute when the difference between the measurements is less than or equal to the `<targetDifference>`
`<measureBlock>`	The block number of the measure block that makes the measurements to be compared; if this is `0` or undefined, the trigger model uses the previous measure block

Details

This block calculates the difference between the last two measurements from a measure block. It subtracts the most recent measurement from the previous measurement.

The difference between the measurements is compared to the target difference. If the difference is less than the target difference, the trigger model goes to the specified branching block. If the difference is more than the target difference, the trigger model proceeds to the next block in the trigger block sequence.

If you do not define the measure block, it will compare measurements of a measure block that precedes the branch delta block. For example, if you have a measure block, a wait block, another measure block, another wait block, and then the branch delta block, the delta block compares the measurements from the second measure block. If a preceding measure block does not exist, an error occurs.

Example

`TRIG:BLOC:BRAN:DELT 5, 0.5, 7, 4`	Configure trigger block 5 to compare the differences between the measurements made in block 4. If the difference between them is less the 0.5, branch to block 7.

Also see

Delta block (on page 3-103)

:TRIGger:BLOCk:BRANch:EVENt

This command branches to a specified block when a specified trigger event occurs.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:EVENt <blockNumber>, <event>, <branchToBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<event>`	The event that must occur before the trigger model branches the specified block
`<branchToBlock>`	The block number of the trigger model block to execute when the specified event occurs

Details

The branch-on-event block goes to a branching block after a specified trigger event occurs. If the trigger event has not yet occurred when the trigger model reaches the branch-on-event block, the trigger model continues to execute the blocks in the normal sequence. After the trigger event occurs, the next time the trigger model reaches the branch-on-event block, it goes to the branching block.

If you set the branch event to none, an error is generated when you run the trigger model.

The following table shows the constants for the events.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	`LAN<n>`
Trigger event blender <n> (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer <n> (1 to 4) expired	`TIMer<n>`
Source limit condition occurs	`SLIMit`

Example

```
:TRIG:BLOC:BRAN:EVEN 6, DISP, 2
```

When the trigger model reaches this block, if the front-panel TRIGGER key has been pressed, the trigger model returns to block 2. If the TRIGGER key has not been pressed, the trigger model continues to block 7 (the next block in the trigger model).

Also see

On event block (on page 3-101)

:TRIGger:BLOCk:BRANch:LIMit:CONStant

This command defines a trigger model block that goes to a specified block if a measurement meets preset criteria.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:LIMit:CONStant <blockNumber>, <limitType>, <limitA>,
   <limitB>, <branchToBlock>
:TRIGger:BLOCk:BRANch:LIMit:CONStant <blockNumber>, <limitType>, <limitA>,
   <limitB>, <branchToBlock>, <measureBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<limitType>`	The type of limit (`ABOVe`, `BELow`, `INside`, or `OUTside`)
`<limitA>`	The limit that the measurement is tested against; if *`limitType`* is set to: • `ABOVe`: This value is ignored • `BELow`: The measurement must be below this value • `INside`: The low limit that the measurement is compared against • `OUTside`: The low limit that the measurement is compared against
`<limitB>`	The upper limit that the measurement is tested against; if *`limitType`* is set to: • `ABOVe`: The measurement must be above this value • `BELow`: This value is ignored • `INside`: The high limit that the measurement is compared against • `OUTside`: The high limit that the measurement is compared against
`<branchToBlock>`	The block number of the trigger model block to execute when the measurement meets the defined criteria
`<measureBlock>`	The block number of the measure block that makes the measurements to be compared; if this is `0` or undefined, the trigger model uses the previous measure block

Details

The branch-on-constant-limits block goes to a branching block if a measurement meets the criteria set by this command.

The type of limit can be:

- Above: The measurement is above the value set by limit B; limit A must be set, but is ignored when this type is selected
- Below: The measurement is below the value set by limit A; limit B must be set, but is ignored when this type is selected

- Inside: The measurement is inside the values set by limits A and B; limit A must be the low value and Limit B must be the high value
- Outside: The measurement is outside the values set by limits A and B; limit A must be the low value and Limit B must be the high value

The measurement block must be a measure building block that occurs in the trigger model before the branch-on-constant-limits block. The last measurement from a measure building block is used.

If the limit A is more than the limit B, the values are automatically swapped so that the lesser value is used as the lower limit.

Example

```
TRIGger:BLOCk:BRANch:LIMit:CONStant 5, OUTside, 0.15, 0.65, 8
```

Configure trigger block 5 to check for measurements in the last measure block. If the measurements are outside of the 0.15 and 0.65 limits, branch to block 8.

Also see

Constant Limit block (on page 3-102)

:TRIGger:BLOCk:BRANch:LIMit:DYNamic

This command defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:LIMit:DYNamic <blockNumber>, <limitType>, <limitNumber>,
   <branchToBlock>
:TRIGger:BLOCk:BRANch:LIMit:DYNamic <blockNumber>, <limitType>, <limitNumber>,
   <branchToBlock>, <measureBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<limitType>`	The type of limit (`ABOVe`, `BELow`, `INside`, or `OUTside`)
`<limitNumber>`	The limit number (`1` or `2`)
`<branchToBlock>`	The block number of the trigger model block to execute when the limits are met
`<measureBlock>`	The block number of the measure block that makes the measurements to be compared; if this is `0` or undefined, the trigger model uses the previous measure block

Details

The branch-on-dynamic-limits block defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

When you define this block, you set:

- The type of limit (above, below, inside, or outside the limit values)
- The limit number (you can have 1 or 2 limits)
- The block to go to if the measurement meets the criteria
- The block that makes the measurement that is compared to the limits; the last measurement from that block is used

There are two user-defined limits: limit 1 and limit 2. Both include their own high and low values, which are set using the front-panel Calculations limit settings or through commands. The results of these limit tests are recorded in the reading buffer that accompanies each stored reading.

Limit values are stored in the measure configuration list, so you can use a configuration list to step through different limit values.

The measure block must occur in the trigger model before the branch-on-dynamic-limits block. If no measure block is defined, the measurement from the previous measure block is used. If no previous measure block exists, an error is reported.

Example

`CALC2:LIM1:STAT ON`
`CALC2:LIM1:LOW -5.17`
`CALC2:LIM1:UPP -4.23`
`TRIG:BLOC:BRAN:LIM:DYN 9, IN, 1, 12, 7`
	Set the limits on with a low limit of -5.17 and a high limit of -4.23. Set trigger block 9 to test if the limit is inside those limits based on the measurement reading at block 7. If the measurement is within the limits, go to block 12.

Also see

Dynamic Limits block (on page 3-103)
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] (on page 6-22)
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] (on page 6-24)

:TRIGger:BLOCk:BRANch:ONCE

This command causes the trigger model to branch to a specified building block the first time it is encountered in the trigger model.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

`:TRIGger:BLOCk:BRANch:ONCE <blockNumber>, <branchToBlock>`

`<blockNumber>`	The sequence of the block in the trigger model
`<branchToBlock>`	The block number of the trigger model block to execute when the trigger model first encounters this block

Details

The branch-once building block branches to a specified block the first time trigger model execution encounters the branch-once block. If it is encountered again, the trigger model ignores the block and continues in the normal sequence.

The once block is reset when trigger model execution reaches the idle state. Therefore, the branch-once block always executes the first time the trigger model execution encounters this block.

Example

```
:TRIG:BLOC:BRAN:ONCE 2, 4
```

The first time the trigger model reaches block 2, the trigger model goes to block 4 instead of proceeding to the default sequence of block 3.

Also see

Once block (on page 3-104)
:TRIGger:BLOCk:BRANch:ONCE:EXCLuded (on page 6-158)

:TRIGger:BLOCk:BRANch:ONCE:EXCLuded

This command causes the trigger model to go to a specified building block every time the trigger model encounters it, except for the first time.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BRANch:ONCE:EXCLuded <blockNumber>, <branchToBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<branchToBlock>`	The block number of the trigger model block to execute when the trigger model encounters this block after the first encounter

Details

The branch-once-excluded block is ignored the first time the trigger model encounters it. After the first encounter, the trigger model goes to the specified branching block.

The branch-once-excluded block is reset when the trigger model starts or is placed in idle.

Example

```
:TRIG:BLOC:BRAN:ONCE:EXCL 2, 4
```

When the trigger model reaches block 2 the first time, the trigger model goes to block 3. If the trigger model reaches this block again, the trigger model goes to block 4.

Also see

Once excluded block (on page 3-104)
:TRIGger:BLOCk:BRANch:ONCE (on page 6-157)

:TRIGger:BLOCk:BUFFer:CLEar

This command defines a trigger model block that clears the reading buffer.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:BUFFer:CLEar <blockNumber>
:TRIGger:BLOCk:BUFFer:CLEar <blockNumber>, "<bufferName>"
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bufferName>`	The name of the buffer, which must be an existing buffer; if no buffer is defined, `defbuffer1` is used

Details

When trigger model execution reaches the buffer clear trigger block, the instrument empties the specified reading buffer. The specified buffer can be the default buffer or a buffer that you defined.

If you are clearing a user-defined reading buffer, you must create the buffer before you define this block.

Example

```
TRIG:LOAD "EMPTY"
TRIG:BLOC:BUFF:CLE 1
TRIG:BLOC:MEAS 2
TRIG:BLOC:BRAN:COUN 3, 5, 2
TRIG:BLOC:DEL:CONS 4, 1
TRIG:BLOC:BRAN:COUN 5, 3, 2
```

Reset trigger model settings.
Clear `defbuffer1` at the beginning of the trigger model.
Loop and take 5 readings.
Delay 1 s.
Loop three more times back to block 2.
At end of execution, 15 readings are stored in `defbuffer1`.

Also see

Buffer clear block (on page 3-93)
:TRACe:MAKE (on page 6-130)

:TRIGger:BLOCk:CONFig:NEXT

This command recalls the settings at the next index of a source or measure configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:CONFig:NEXT <blockNumber>, "<configurationList>"
```

`<blockNumber>`	The sequence of the block in the trigger model
`<configurationList>`	The source or measure configuration list from which to recall settings

Details

When trigger model execution reaches a configuration recall next block, the settings at the next index in the specified configuration list are restored.

The first time the trigger model encounters this block for a specific configuration list, the first index is recalled. Each subsequent time this block is encountered, the settings at the next index in the configuration list are recalled and take effect before the next step executes. When the last index in the list is reached, it returns to the first index.

The configuration list must be defined before you can use this block.

Example

`TRIG:BLOC:CONF:NEXT 12, "SETTINGS_LIST"`	Set trigger block 12 to restore the settings from the next index that is stored in the configuration list `SETTINGS_LIST`.

Also see

Configuration lists (on page 3-39)

:TRIGger:BLOCk:CONFig:PREVious

This command defines a trigger model block that recalls the settings stored at the previous index in a source or measure configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:CONFig:PREVious <blockNumber>, "<configurationList>"
```

`<blockNumber>`	The sequence of the block in the trigger model
`<configurationList>`	The source or measure configuration list from which to recall settings

Details

The Config List Prev building block defines a trigger model block that recalls the settings stored at the previous index in a source or measure configuration list.

The configuration list previous index trigger block type recalls the previous index in a configuration list. It configures the source or measure settings of the instrument based on the settings at that index. The trigger model executes the settings at that index before the next block is executed.

The first time the trigger model encounters this block, the last index in the configuration list is recalled. Each subsequent time trigger model execution reaches a configuration list previous block for this configuration list, it goes backward one index. When the first index in the list is reached, it goes to the last index in the configuration list.

You must create the configuration list before you can define it in this building block.

Example

`TRIG:BLOC:CONF:PREV 14, "SETTINGS_LIST"`	Set trigger block 14 to restore the settings from the previous index that is stored in the configuration list `SETTINGS_LIST`.

Also see

Configuration lists (on page 3-39)

:TRIGger:BLOCk:CONFig:RECall

This command recalls the system settings that are stored in a source or measure configuration list.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:CONFig:RECall <blockNumber>, "<configurationList>"
:TRIGger:BLOCk:CONFig:RECall <blockNumber>, "<configurationList>", <index>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<configurationList>`	A string that defines the configuration list to recall
`<index>`	The index in the configuration list to recall; defaults to 1 if not selected

Details

When the trigger model reaches a configuration recall building block, the settings in the specified configuration list are recalled.

You can restore a specific set of configuration settings in the configuration list by defining the index.

Example

`SOUR:CONF:LIST:CRE "biasLevel"`
`SOUR:FUNC VOLT`
`SENS:FUNC "CURR"`
`SOUR:VOLT:LEV 5`
`SOUR:CONF:LIST:STORE "biasLevel"`
`TRIG:BLOCK:CONF:RECALL 1, "biasLevel", 1`
	Create a configuration list named `biasLevel`. Set the source function to 5 V and the measure function to current.
	Store the source settings at index 1 in the configuration list named `biasLevel`.
	Recall index 1 of the configuration list named `biasLevel` as block 1 of the trigger model.

Also see

Configuration lists (on page 3-39)
:SOURce[1]:CONFiguration:LIST:STORe (on page 6-75)

:TRIGger:BLOCk:DELay:CONStant

This command adds a constant delay to the trigger model.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:DELay:CONStant <blockNumber>, <time>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<time>`	The amount of time to delay (167 ns to 10,000 s, or 0 for no delay)

Details

When trigger model execution reaches a delay block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made, and if any previously executed block started infinite measurements, they also continue to be made.

This delay waits for the set amount of delay time to elapse before proceeding to the next block in the trigger model.

If other delays have been set, this delay is in addition to the other delays.

Example

`SOUR:CONF:LIST:CRE "ampLevel"`	Create configuration lists named `ampLevel` and `biasLevel`.
`SOUR:CONF:LIST:CRE "biasLevel"`	
`SOUR:FUNC VOLT`	Set the source function to 5 V and the measurement function to current.
`SENS:FUNC "CURR"`	
`SOUR:VOLT:LEV 5`	Store the settings in the `ampLevel` configuration list at index 1.
`SOUR:CONF:LIST:STORE "ampLevel"`	
`SOUR:VOLT:LEV 0`	Set the voltage level to 0 V.
`SOUR:CONF:LIST:STORE "biasLevel"`	Store the setting in the `biasLevel` configuration list at index 1.
`TRIG:BLOC:SOUR:STATE 1, ON`	Set block 1 to turn the source output on.
`TRIG:BLOCK:CONF:RECALL 2, "ampLevel", 1`	Set block 2 to recall the `ampLevel` settings from index 1.
`TRIG:BLOC:DEL:CONS 3, 0.1`	Set block 3 to provide a constant delay of 0.1 s.
`TRIG:BLOCK:MEAS 4`	Set block 4 to make a measurement.
`TRIG:BLOCK:CONF:RECALL 5, "biasLevel", 1`	Set block 5 to recall the `biasLevel` settings at index 1.
`TRIG:BLOCK:DEL:CONS 6, 0.2`	Set block 6 to provide a constant delay of 0.2 s.
`TRIG:BLOCK:BRAN:COUN 7,19,2`	Set block 7 to repeat block 2 nineteen times.
`INIT`	

Also see

None

:TRIGger:BLOCk:DELay:DYNamic

This command adds a delay to the execution of the trigger model.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:DELay:DYNamic <blockNumber>, <userDelay>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<userDelay>`	The number of the user delay: • `SOURce<n>`, where `<n>` is the number of the user delay (1 to 5) set by `:SOURce[1]:<function>:DELay:USER<n>` • `MEASure<n>`, where `<n>` is the number of the user delay (1 to 5) set by `[:SENSe[1]]:<function>:DELay:USER<n>`

Details

When trigger model execution reaches a dynamic delay block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made.

Each measure function can have up to 5 unique user delay times (M1 to M5). Each source function can also have up to 5 unique user delay times (S1 to S5). The delay time is set by the user-delay command, which is only available over a remote interface.

Example

`:SOUR:VOLT:DEL:USER1 5`	Set user delay 1 for the voltage source to 5 s.
`:TRIG:BLOC:SOUR:STAT 1, ON`	Set trigger block 1 to turn the source output on.
`:TRIG:BLOC:DEL:DYN 2, SOUR1`	Set trigger block 2 to a dynamic delay that calls source user delay 1.
`:TRIG:BLOC:MEAS 3`	Set trigger block 3 to make a measurement.
`:TRIG:BLOC:SOUR:STAT 4, OFF`	Set trigger block 4 to turn the source output off.
`:TRIG:BLOC:BRAN:COUN 5, 10, 1`	Set trigger block 5 to branch to block 1 ten times.
`:INIT`	Start the trigger model.

Also see

[:SENSe[1]]:<function>:DELay:USER<n> (on page 6-50)
:SOURce[1]:<function>:DELay:USER<n> (on page 6-78)

:TRIGger:BLOCk:DIGital:IO

This command defines a trigger model block that sets the lines on the digital I/O port high or low.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:DIGital:IO <blockNumber>, <bitPattern>
:TRIGger:BLOCk:DIGital:IO <blockNumber>, <bitPattern>, <bitMask>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bitPattern>`	Sets the value that specifies the output line bit pattern (0 to 63)
`<bitMask>`	Specifies the bit mask; if omitted, all lines are driven (0 to 63)

Details

To set the lines on the digital I/O port high or low, you can send a bit pattern that is specified as an integer value. The least significant bit maps to digital I/O line 1 and the most significant bit maps to digital I/O line 6.

The bit mask defines the bits in the pattern that are driven high or low. A binary 1 in the bit mask indicates that the corresponding I/O line should be driven according to the bit pattern. To drive all lines, specify all ones (63) or omit this parameter. If the bit for a line in the bit pattern is set to 1, the line is driven high. If the bit is set to 0 in the bit pattern, the line is driven low.

For this block to work as expected, make sure you configure the trigger type and line state of the digital line for use with the trigger model (use the digital line mode command).

Example

```
:DIGital:LINE3:MODE DIG,OUT
:DIGital:LINE4:MODE DIG,OUT
:DIGital:LINE5:MODE DIG,OUT
:DIGital:LINE6:MODE DIG,OUT
:TRIG:BLOC:DIG:IO 4, 20, 60
```

The first four lines of code configures digital I./O lines 3 through 6 as digital outputs.
Trigger block `4` is then configured with a bit pattern of `20` (digital I/O lines 3 and 5 high). The optional bit mask is specified as `60` (lines 3 through 6), so both lines 3 and 5 are driven high.

Also see

:DIGital:LINE<n>:MODE (on page 6-26)
Digital I/O bit weighting (on page 3-80)
Digital I/O port configuration (on page 3-72)

:TRIGger:BLOCk:LIST?

This command returns the settings for all trigger model blocks.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:BLOCk:LIST?
```

Details

This returns the settings for the trigger model.

Example

`:TRIG:BLOC:LIST?`	Returns the settings for the trigger model. Example output is:
	`1) BUFFER_CLEAR BUFFER: defbuffer1`
	`2) MEASURE BUFFER: defbuffer1 COUNT: 1`
	`3) BRANCH_COUNTER VALUE: 5 BRANCH_BLOCK: 2`
	`4) DELAY_CONSTANT DELAY: 1.000000000`
	`5) BRANCH_COUNTER VALUE: 3 BRANCH_BLOCK: 2`

Also see

None

:TRIGger:BLOCk:LOG:EVENt

This command allows you to log an event in the event log when the trigger model is running.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:LOG:EVENt <blockNumber>, <eventNumber>, "<message>"
```

`<blockNumber>`	The sequence of the block in the trigger model
`<eventNumber>`	The event number: • `INFO<n>` • `WARNing<n>` • `ERRor<n>` Where <n> is 1 to 4; you can define up to four of each type You can also set `ABORt`, which aborts the trigger model immediately and posts a warning event log message
`<message>`	A string up to 31 characters

Details

This block allows you to log an event in the event log when trigger model execution reaches this block. You can also force the trigger model to abort with this block. When the trigger model executes the block, the defined event is logged. If the abort option is selected, the trigger model is also aborted immediately.

You can define the type of event (information, warning, abort model, or error). All events generated by this block are logged in the event log. Warning and error events are also displayed in a popup on the front-panel display.

Note that using this block too often in a trigger model could overflow the event log. It may also take away from the time needed to process more critical trigger model blocks.

Example

`TRIGger:BLOCk:LOG:EVENt 9, INFO2, "Trigger model complete"`	Set trigger model block 9 to log an event when the trigger model completes.

Also see

None

:TRIGger:BLOCk:MEASure

This command defines a trigger block that makes a measurement.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:MEASure <blockNumber>
:TRIGger:BLOCk:MEASure <blockNumber>, "<bufferName>"
:TRIGger:BLOCk:MEASure <blockNumber>, "<bufferName>", <count>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bufferName>`	The name of the buffer, which must be an existing buffer; if no buffer is defined, `defbuffer1` is used
`<count>`	Specifies the number of readings to make before moving to the next block in the trigger model; set to a specific value or to infinite by setting to `INF`; to stop the count, set to `0`; if not specified, defaults to `1`

Details

When trigger model execution reaches the block:

1. The instrument begins making a measurement.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurement into the specified reading buffer, which cannot be of the writable buffer style.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and measurements continue in the background until the trigger model execution reaches another measure block or until the trigger model ends.

Example

`TRIG:LOAD "EMPTY"`
`TRIG:BLOC:BUFF:CLEAR 1, "defbuffer2"`
`TRIG:BLOC:MEAS 2, "defbuffer2"`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
	Reset trigger model settings.
	Clear `defbuffer2` at the beginning of the trigger model. Set the measurements to be stored in `defbuffer2`.
	Loop and take 5 readings.
	Delay 1 s.
	Loop three more times back to block 2.
	At end of execution, 15 readings are stored in `defbuffer2`.

Also see

Measure block (on page 3-93)
:TRACe:MAKE (on page 6-130)

:TRIGger:BLOCk:NOP

This command creates a placeholder that performs no action in the trigger model; available only using remote commands.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:NOP <blockNumber>
```

`<blockNumber>`	The sequence of the block in the trigger model

Details

If you remove a trigger model block, you can use this block as a placeholder for the block number so that you do not need to renumber the other blocks.

Example

```
TRIG:BLOC:NOP 5
```
Set block number 5 to be a no operation block.

Also see

None

:TRIGger:BLOCk:NOTify

This command defines a trigger model block that generates a trigger event and immediately continues to the next block.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:NOTify <blockNumber>, <notifyID>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<notifyID>`	The identification number of the notification; 1 to 8

Details

When trigger model execution reaches a notify block, the instrument generates a trigger event and immediately continues to the next block.

Other commands can reference the event that the notify block generates. This assigns a stimulus somewhere else in the system. For example, you can use the notify event as the stimulus of a hardware trigger line, such as a digital I/O line.

When you call this event, you use the format NOTIFY followed by the notify identification number. For example, if you assign <notifyID> as 4, you would refer to it as NOTIFY4 in the command that references it.

Example

`:TRIG:BLOC:NOT 5, 2` `:TRIG:BLOC:BRAN:EVEN 6, NOTIFY2, 2`	Define trigger model block 5 to be the notify 2 event. Assign the notify 2 event to be the trigger for stimulus for the branch event for block 6.

Also see

Notify block (on page 3-96)

:TRIGger:BLOCk:SOURce:STATe

This command defines a trigger block that turns the output source on or off.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:SOURce:STATe <blockNumber>, <state>
```

<blockNumber>	The sequence of the block in the trigger model
<state>	Disable the source: OFF Enable the source: ON

Details

The source output block determines if the output source is turned on or off when the trigger model reaches this block.

This block does not determine the settings of the output source (such as the output voltage level and source delay). The source settings are determined by either the present settings of the instrument or by a source configuration list.

When you list trigger blocks, this block is listed as SOURCE_OUTPUT.

Example

`TRIG:BLOC:SOUR:STAT 1, 1`
`TRIG:BLOC:DEL:CONS 2, 0.01`
`TRIG:BLOC:MEAS 3`
`TRIG:BLOC:BRAN:COUN 4, 100, 2`
`TRIG:BLOC:SOUR:STAT 5, 0`
	This example turns the output on.
	Delay 10 ms.
	Make a measurement.
	Loop and take 100 readings.
	The output is turned off after the 100 readings are made.

Also see

Wait block (on page 3-94)

:TRIGger:BLOCk:WAIT

This command defines a trigger model block that waits for an event before allowing the trigger model to continue.

Type	Affected by	Where saved	Default value
Command only	Recall settings Instrument reset Power cycle	Save settings	Not applicable

Usage

```
:TRIGger:BLOCk:WAIT <blockNumber>, <event>
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <clear>
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <clear>, <logic>, <event>
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <clear>, <logic>, <event>, <event>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<event>`	The event that must occur before the trigger block allows trigger execution to continue; see **Details** for event names
`<clear>`	To clear previously detected trigger events when entering the wait block: `ENTer` To immediately act on any previously detected triggers and not clear them (default): `NEVer`
`<logic>`	If each event must occur before the trigger model continues: `AND` If at least one of the events must occur before the trigger model continues: `OR`

Details

You can use the wait block to synchronize measurements with other instruments and devices.

You can set the instrument to wait for the following events:

- Front-panel TRIGGER key press
- Notify (only available when using remote commands)
- Command interface trigger
- Digital input/output signals, such as DIGIO and TSP-Link
- LAN
- Blender
- Timer
- Source limit condition

The event can occur before trigger model execution reaches the wait block. If the event occurs after trigger model execution starts but before the trigger model execution reaches the wait block, the trigger model records the event. By default, when trigger model execution reaches the wait block, it executes the wait block without waiting for the event to happen again (the clear parameter is set to never).

The instrument clears the memory of the recorded event when trigger model execution is at the start block and when the trigger model exits the wait block. It also clears the recorded trigger event when the clear parameter is set to enter.

All items in the list are subject to the same action; you cannot combine `AND` and `OR` logic in a single block.

You cannot leave the first event as no trigger. If the first event is not defined, the trigger model errors when you attempt to initiate it.

The following usage has been deprecated; replace it with the usage above that includes the *clear* parameter.

```
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <logic>, <event>
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <logic>, <event>, <event>
```

The following table shows the constants for the events.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	`LAN<n>`
Trigger event blender <n> (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer <n> (1 to 4) expired	`TIMer<n>`
Source limit condition occurs	`SLIMit`

Example 1

`:TRIGger:BLOCk:WAIT 9, DISP`	Set trigger model block 9 to wait for a user to press the TRIGGER key on the front panel before continuing and to act on a recorded TRIGGER key event that gets detected either before or after reaching block 9.

Example 2

`:TRIGger:BLOCk:WAIT 9, DISP, ENTer`	Set trigger model block 9 to wait for a user to press the TRIGGER key on the front panel before continuing and to act only on a recorded TRIGGER key event that gets detected when block 9 is reached.

Also see

Wait block (on page 3-94)

:TRIGger:DIGital<n>:IN:CLEar

This command clears the trigger event on a digital input line.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:DIGital<n>:IN:CLEar
```

<n>	Digital I/O trigger line (1 to 6)

Details

The event detector of a trigger enters the detected state when an event is detected. For the specified trigger line, this command clears the event detector, discards the history, and clears the overrun status (sets the overrun status to 0).

For this block to work as expected, make sure you configure the trigger type and line state of the digital line for use with the trigger model (use the digital line mode command).

Example

`:TRIG:DIG2:IN:CLE`	Clears the trigger event detector on I/O line 2.

Also see

:DIGital:LINE<n>:MODE (on page 6-26)
Digital I/O port configuration (on page 3-72)
:TRIGger:DIGital<n>:IN:OVERrun? (on page 6-173)

:TRIGger:DIGital<n>:IN:EDGE

This command sets the edge used by the trigger event detector on the given trigger line.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	FALL

Usage

```
:TRIGger:DIGital<n>:IN:EDGE <detectedEdge>
:TRIGger:DIGital<n>:IN:EDGE?
```

<n>	Digital I/O trigger line (1 to 6)
<detectedEdge>	The trigger edge value: • Detect falling-edge triggers as inputs: FALLing • Detect rising-edge triggers as inputs: RISing • Detect either falling or rising-edge triggers as inputs: EITHer See **Details** for descriptions of values

Details

This command sets the logic on which the trigger event detector and the output trigger generator operate on the specified trigger line.

To directly control the line state, set the mode of the line to digital and use the write command. When the digital line mode is set for open drain, the edge settings assert a TTL low-pulse.

Trigger mode values

Value	Description
FALLing	Detects falling-edge triggers as input when the line is configured as an input or open drain.
RISing	Detects rising-edge triggers as input when the line is configured as an open drain.
EITHer	Detects rising- or falling-edge triggers as input when the line is configured as an input or open drain.

Example

```
:DIG:LINE4:MODE TRIG,IN
:TRIG:DIG4:IN:EDGE RIS
```

Sets the input trigger mode for the digital I/O line 4 to detect rising-edge triggers as input.

Also see

Digital I/O port configuration (on page 3-72)
:DIGital:LINE<n>:MODE (on page 6-26)
:DIGital:WRITe <n> (on page 6-30)
:TRIGger:DIGital<n>:IN:CLEar (on page 6-172)

:TRIGger:DIGital<n>:IN:OVERrun?

This command returns the event detector overrun status.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:DIGital<n>:IN:OVERrun?
```

<n>	Digital I/O trigger line (1 to 6)

Details

This command returns the event detector overrun status as 0 (false) or 1 (true).

If this is 1, an event was ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the line itself. It does not indicate if an overrun occurred in any other part of the trigger model or in any other detector that is monitoring the event.

Example

`TRIG:DIG1:IN:OVER?`	Returns `0` if no overruns have occurred or `1` if one or more overrun have occurred for I/O line 1.

Also see

Digital I/O port configuration (on page 3-72)
:DIGital:LINE<n>:MODE (on page 6-26)

:TRIGger:DIGital<n>:OUT:LOGic

This command sets the output logic of the trigger event generator to positive or negative for the specified line.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	NEG

Usage

```
:TRIGger:DIGital<n>:OUT:LOGic <logicType>
:TRIGger:DIGital<n>:OUT:LOGic?
```

`<n>`	Digital I/O trigger line (1 to 6)
`<logicType>`	The output logic of the trigger generator: • Assert a TTL-high pulse for output: `POSitive` • Assert a TTL-low pulse for output: `NEGative`

Details

This command sets the trigger event generator to assert a TTL pulse for output logic. Positive is a high pulse; negative is a low pulse.

Example

`:DIG:LINE4:MODE TRIG, OUT` `:TRIG:DIG4:OUT:LOG NEG`	Sets line 4 mode to be a trigger output and sets the output logic of the trigger event generator to negative (asserts a low pulse).

Also see

:DIGital:LINE<n>:MODE (on page 6-26)
Digital I/O port configuration (on page 3-72)

:TRIGger:DIGital<n>:OUT:PULSewidth

This command describes the length of time that the trigger line is asserted for output triggers.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	10e-6 (10 µs)

Usage

```
:TRIGger:DIGital<n>:OUT:PULSewidth <width>
:TRIGger:DIGital<n>:OUT:PULSewidth?
```

<n>	Digital I/O trigger line (1 to 6)
<width>	Pulse length (0 to 100,000 s)

Details

Setting the pulse width to zero (0) seconds asserts the trigger indefinitely.

Example

```
DIG:LINE1:MODE TRIG, OUT
TRIG:DIG1:OUT:PULS 2
```

Set digital line 1 to trigger out.
Set the pulse to 2 s.

Also see

:DIGital:LINE<n>:MODE (on page 6-26)
:DIGital:WRITe <n> (on page 6-30)
Digital I/O port configuration (on page 3-72)

:TRIGger:DIGital<n>:OUT:STIMulus

This command selects the event that causes a trigger to be asserted on the digital output line.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	NONE

Usage

```
:TRIGger:DIGital<n>:OUT:STIMulus <event>
:TRIGger:DIGital<n>:OUT:STIMulus?
```

`<n>`	Digital I/O trigger line (1 to 6)
`<event>`	The event to use as a stimulus; see **Details**

Details

The digital trigger pulsewidth command determines how long the trigger is asserted.

The trigger stimulus for a digital I/O line can be set to one of the trigger events that are described in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	`LAN<n>`
Trigger event blender <n> (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer <n> (1 to 4) expired	`TIMer<n>`
Source limit condition occurs	`SLIMit`

Example

`:TRIG:DIG2:OUT:STIMulus TIM3`	Set the stimulus for output digital trigger line 2 to be the expiration of trigger timer 3.

Also see

Digital I/O port configuration (on page 3-72)
:DIGital:LINE<n>:STATe (on page 6-28)
:TRIGger:DIGital<n>:OUT:LOGic (on page 6-174)

:TRIGger:LAN<n>:IN:CLEar

This command clears the event detector for a LAN trigger.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LAN<n>:IN:CLEar
```

`<n>`	The LAN event number (1 to 8) to clear

Details

The trigger event detector enters the detected state when an event is detected. This function clears a trigger event detector and discards the previous of the trigger packet.

This function clears all overruns associated with this LAN trigger.

Example

`:TRIG:LAN5:IN:CLE`	Clears the event detector with LAN packet 5.

Also see

:TRIGger:LAN<n>:IN:OVERrun? (on page 6-178)

:TRIGger:LAN<n>:IN:EDGE

This command sets the trigger operation and detection mode of the specified LAN event.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	EITH

Usage

```
:TRIGger:LAN<n>:IN:EDGE <mode>
:TRIGger:LAN<n>:IN:EDGE?
```

`<n>`	The LAN event number (1 to 8)
`<mode>`	The trigger mode; see the **Details** for more information

Details

This command controls how the trigger event detector and the output trigger generator operate on the given trigger. These settings are intended to provide behavior similar to the digital I/O triggers.

LAN trigger mode values		
Mode	**Trigger packets detected as input**	**LAN trigger packet generated for output with a…**
`EITHer`	Rising or falling edge (positive or negative state)	negative state
`FALLing`	Falling edge (negative state)	negative state
`RISing`	Rising edge (positive state)	positive state

Example

`:TRIG:LAN2:IN:EDGE FALL`	Set the LAN trigger mode for event 2 to falling.

Also see

Digital I/O (on page 3-70)
TSP-Link System Expansion Interface (on page 3-133)

:TRIGger:LAN<n>:IN:OVERrun?

This command indicates the overrun status of the LAN event detector.

Type	Affected by	Where saved	Default value
Query only	TRIGger:LAN<n>:IN:CLEar	Not applicable	Not applicable

Usage

```
:TRIGger:LAN<n>:IN:OVERrun?
```

`<n>`	The LAN event number (1 to 8)

Details

This command indicates whether an event has been ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the synchronization line itself. It does not indicate if an overrun occurred in any other part of the trigger model, or in any other construct that is monitoring the event.

It also is not an indication of an output trigger overrun.

The trigger overrun state for the specified LAN packet is returned as `1` (true) or `0` (false).

Example

`TRIG:LAN5:IN:OVER?`	Checks the overrun status of a trigger on LAN5 and outputs the value, such as: `0`

Also see

:TRIGger:LAN<n>:IN:CLEar (on page 6-177)

:TRIGger:LAN<n>:OUT:CONNect:STATe

This command prepares the event generator for outgoing trigger events.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LAN<n>:OUT:CONNect:STATe <state>
:TRIGger:LAN<n>:OUT:CONNect:STATe?
```

`<n>`	The LAN event number (1 to 8)
`<state>`	Do not send event messages: `OFF` or `0` Prepare to send event messages: `ON` or `1`

Details

When this is set to `ON`, the instrument prepares the event generator to send event messages. For TCP connections, this opens the TCP connection.

The event generator automatically disconnects when either the protocol or IP address for this event is changed.

When this is set to `OFF`, for TCP connections, this closes the TCP connection.

Example

`:TRIGger:LAN1:OUT:PROTocol MULT` `:TRIGger:LAN1:OUT:CONNect:STATe ON`	Set the protocol to multicast and prepare the event generator to send event messages.

Also see

:TRIGger:LAN<n>:OUT:IP:ADDRess (on page 6-179)
:TRIGger:LAN<n>:OUT:PROTocol (on page 6-181)

:TRIGger:LAN<n>:OUT:IP:ADDRess

This command specifies the address (in dotted-decimal format) of UDP or TCP listeners.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	"0.0.0.0"

Usage

```
:TRIGger:LAN<n>:OUT:IP:ADDRess "<address>"
:TRIGger:LAN<n>:OUT:IP:ADDRess?
```

`<n>`	The LAN event number (1 to 8)
`<address>`	A string that represents the LAN address in dotted decimal notation

Details

Sets the IP address for outgoing trigger events.

After you change this setting, you must send the connect command before outgoing messages can be sent.

Example

`TRIG:LAN1:OUT:IP:ADDR "192.0.32.10"`	Use IP address `192.0.32.10` to connect the LAN trigger.

Also see

:TRIGger:LAN<n>:OUT:CONNect:STATe (on page 6-179)

:TRIGger:LAN<n>:OUT:LOGic

This command sets the logic on which the trigger event detector and the output trigger generator operate on the given trigger line.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	NEG

Usage

```
:TRIGger:LAN<n>:OUT:LOGic <logicType>
:TRIGger:LAN<n>:OUT:LOGic?
```

`<n>`	The LAN event number (1 to 8)
`<logicType>`	The type of logic: • `POSitive` • `NEGative`

Example

`TRIG:LAN1:OUT:LOG POS`	Set the logic to positive.

Also see

None

:TRIGger:LAN<n>:OUT:PROTocol

This command sets the LAN protocol to use for sending trigger messages.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	TCP

Usage

```
:TRIGger:LAN<n>:OUT:PROTocol
:TRIGger:LAN<n>:OUT:PROTocol?
```

`<n>`	The LAN event number (1 to 8)
`<protocol>`	The protocol to use for messages from the trigger: • `TCP` • `UDP` • `MULTicast`

Details

The LAN trigger listens for trigger messages on all the supported protocols. However, it uses the designated protocol for sending outgoing messages.

After you change this setting, you must re-connect the LAN trigger event generator before you can send outgoing event messages.

When multicast is selected, the trigger IP address is ignored and event messages are sent to the multicast address 224.0.23.159.

Example

`:TRIG:LAN1:OUT:PROT TCP` `:TRIG:LAN1:OUT:CONN:STAT`	Set the LAN protocol for trigger messages to be TCP and re-connect the LAN trigger event generator.

Also see

:TRIGger:LAN<n>:OUT:CONNect:STATe (on page 6-179)
:TRIGger:LAN<n>:OUT:IP:ADDRess (on page 6-179)

:TRIGger:LAN<n>:OUT:STIMulus

This command specifies events that cause this trigger to assert.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	NONE

Usage

```
:TRIGger:LAN<n>:OUT:STIMulus <LANevent>
:TRIGger:LAN<n>:OUT:STIMulus?
```

`<n>`	A number specifying the trigger packet over the LAN for which to set or query the trigger source (1 to 8)
`<LANevent>`	The LAN event that causes this trigger to assert

Details

This attribute specifies which event causes a LAN trigger packet to be sent for this trigger. Set the event to one of the existing trigger events, which are shown in the following table.

Setting this attribute to none disables automatic trigger generation.

If any events are detected before the trigger LAN connection is sent, the event is ignored and the action overrun is set.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	`LAN<n>`
Trigger event blender <n> (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer <n> (1 to 4) expired	`TIMer<n>`
Source limit condition occurs	`SLIMit`

Example

`TRIG:LAN1:OUT:STIM TIM1`	Set the timer 1 trigger event as the source for the LAN packet 1 trigger stimulus.

Also see

:TRIGger:LAN<n>:OUT:CONNect:STATe (on page 6-179)

:TRIGger:LOAD "ConfigList"

This command loads a predefined trigger model configuration that uses source and measure configuration lists.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "ConfigList", "<measureConfigList>", "<sourceConfigList>"
:TRIGger:LOAD "ConfigList", "<measureConfigList>", "<sourceConfigList>", <delay>
:TRIGger:LOAD "ConfigList", "<measureConfigList>", "<sourceConfigList>", <delay>,
   "<bufferName>"
```

`<measureConfigList>`	A string that contains the name of the measurement configuration list to use
`<sourceConfigList>`	A string that contains the name of the source configuration list to use
`<delay>`	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This trigger model template incorporates a source configuration list and measure configuration list. You must set up the configuration lists before loading the trigger model.

You can also set a delay and change the reading buffer.

After selecting a trigger model template, you can view the trigger model blocks in a graphical format by pressing the front-panel **MENU** key and under Trigger, selecting **Configure**. You can also add or delete blocks and change trigger model settings from this screen. You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

This command replaces the `TRIGger:LOAD:CONFiguration:LIST` command, which is deprecated.

Example

```
*RST
:SOURce:CONF:LIST:CRE "SOURCE_LIST"
:SENS:CONF:LIST:CRE "MEASURE_LIST"
:SOUR:VOLT 1
:SOURce:CONF:LIST:STORE "SOURCE_LIST"
:SENS:CURR:RANG 1e-3
:SENSe:CONF:LIST:STOR "MEASURE_LIST"
:SOUR:VOLT 5
:SOURce:CONF:LIST:STORE "SOURCE_LIST"
:SENS:CURR:RANG 10e-3
:SENSe:CONF:LIST:STOR "MEASURE_LIST"
:SOUR:VOLT 10
:SOURce:CONF:LIST:STORE "SOURCE_LIST"
:SENS:CURR:RANG 100e-3
:SENSe:CONF:LIST:STOR "MEASURE_LIST"
:TRIG:LOAD "ConfigList", "MEASURE_LIST", "SOURCE_LIST"
INIT
```

Set up a source configuration list named `SOURCE_LIST` and a measurement configuration list named `MEASURE_LIST`.
Create the source list with three indexes that set the source voltage levels to 1, 5, and 10.
Create the measure list with three indexes that set the measure current ranges to 1 mA, 10 mA, and 100 mA.
Load the configuration list trigger model, using these two configuration lists. Start the trigger model.

Also see

None

:TRIGger:LOAD "DurationLoop"

This command loads a predefined trigger model configuration that makes continuous measurements for a specified amount of time.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "DurationLoop", <duration>
:TRIGger:LOAD "DurationLoop", <duration>, <delay>
:TRIGger:LOAD "DurationLoop", <duration>, <delay>, "<readingBuffer>"
```

`<duration>`	The amount of time for which to make measurements (167 ns to 100 ks)
`<delay>`	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
`<readingBuffer>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

When you load this predefined trigger model, you can specify amount of time to make a measurement and the length of the delay before the measurement.

After selecting a trigger model template, you can view the trigger model blocks in a graphical format by pressing the front-panel **MENU** key and under Trigger, selecting **Configure**. You can also add or delete blocks and change trigger model settings from this screen. You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

This command replaces the `TRIGger:LOAD:LOOP:DURation` command, which is deprecated.

Example

`*RST`
`SOUR:FUNC VOLT`
`SOUR:VOLT 5`
`SENS:FUNC "CURR"`
`TRIG:LOAD "DurationLoop", 10, 0.01`
`INIT`
	Reset the instrument. Set the instrument to source voltage at 5 V. Set to measure current.
	Load the Duration Loop trigger model to take measurements for 10 s with a 10 ms delay before each measurement.
	Start the trigger model.

Also see

None

:TRIGger:LOAD "Empty"

This command resets the trigger model.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "Empty"
```

Details

When you load this predefined trigger model, any blocks that have been defined in the trigger model are cleared sothe trigger model has no blocks defined.

This command replaces the `TRIGger:LOAD:EMPTy` command, which is deprecated.

Example

`TRIG:LOAD "Empty"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of execution of the trigger model.
	Loop and take 5 readings.
	Delay 1 s.
	Loop three more times back to block 2.
	At the end of execution, 15 readings are stored in `defbuffer1`.

Also see

None

:TRIGger:LOAD "GradeBinning"

This command loads a predefined trigger model configuration that sets up a grading operation.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
   <limit3Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
   <limit3Pattern>, <limit4High>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
   <limit3Pattern>, <limit4High>, <limit4Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
   <limit3Pattern>, <limit4High>, <limit4Low>, <limit4Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
   <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
   <limit2High>, <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>,
   <limit3Pattern>, <limit4High>, <limit4Low>, <limit4Pattern>, "<bufferName>"
```

<components>	The number of components to measure (1 to 268,435,455)
<startInLine>	The input line that starts the test; 5 for digital line 5, 6 for digital line 6; default is 5
<startDelay>	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
<endDelay>	The delay time after the measurement (167 ns to 10 ks); default is 0 for no delay

<limit*x*High>	*x* is limit 1, 2, 3, or 4; the upper limit that the measurement is compared against
<limit*x*Low>	*x* is 1, 2, 3, or 4; the lower limit that the measurement is compared against
<limit1Pattern>	The bit pattern that is sent when the measurement fails limit 1; range 1 to 15; default is 1
<limit2Pattern>	The bit pattern that is sent when the measurement fails limit 2; range 1 to 15; default is 2
<limit3Pattern>	The bit pattern that is sent when the measurement fails limit 3; range 1 to 15; default is 4
<limit4Pattern>	The bit pattern that is sent when the measurement fails limit 4; range 1 to 15; default is 8
<allPattern>	The bit pattern that is sent when all limits have passed; 1 to 15; default is 15
<bufferName>	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This trigger model template allows you to grade components and place them into up to four bins, based on the comparison to limits.

To set a limit as unused, set the high value for the limit to be less than the low limit.

All limit patterns and the pass pattern are sent on digital I/O lines 1 to 4, where 1 is the least significant bit.

After selecting a trigger model template, you can view the trigger model blocks in a graphical format by pressing the front-panel **MENU** key and under Trigger, selecting **Configure**. You can also add or delete blocks and change trigger model settings from this screen. You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

Also see

None

:TRIGger:LOAD "LogicTrigger"

This command loads a predefined trigger model configuration that sets up a digital trigger through the digital I/O.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "LogicTrigger", <digInLine>, <digOutLine>, <count>
:TRIGger:LOAD "LogicTrigger", <digInLine>, <digOutLine>, <count>, <clear>
:TRIGger:LOAD "LogicTrigger", <digInLine>, <digOutLine>, <count>, <clear>, <delay>
:TRIGger:LOAD "LogicTrigger", <digInLine>, <digOutLine>, <count>, <clear>, <delay>,
   "<bufferName>"
```

`<digInLine>`	The digital input line (1 to 6); also the event that the trigger model will wait on in block 1
`<digOutLine>`	The digital output line (1 to 6)
`<count>`	The number of measurements the instrument will make
`<clear>`	To clear previously detected trigger events when entering the wait block: `ENTer` To immediately act on any previously detected triggers and not clear them (default): `NEVer`
`<delay>`	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
`<bufferName>`	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; default is `defbuffer1`

Details

This trigger model waits for a digital input event to occur, makes a measurement, and issues a notify event. The notify event asserts a digital output line.

After selecting a trigger model template, you can view the trigger model blocks in a graphical format by pressing the front-panel **MENU** key and under Trigger, selecting **Configure**. You can also add or delete blocks and change trigger model settings from this screen. You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

This command replaces the `:TRIGger:LOAD:TRIGger:EXTernal` command, which is deprecated.

Example

```
:TRIGger:LOAD "LogicTrigger", 1, 2, 10, ENTer, 0.001, "defbuffer1"
```

Set up the template to use the digital in line 1 and wait until the wait block is entered to detect the pulse from digital in line 1 and to trigger measurements.
Pulse digital out line 2 when the measurement is complete.
Make 10 measurements, with a delay of 1 ms before each measurement.
Store the measurements in `defbuffer1`.

Also see

None

:TRIGger:LOAD "LoopUntilEvent"

This command loads a predefined trigger model configuration that makes continuous measurements until the specified event occurs.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "LoopUntilEvent", <eventConstant>, <position>
:TRIGger:LOAD "LoopUntilEvent", <eventConstant>, <position>, <clear>
:TRIGger:LOAD "LoopUntilEvent", <eventConstant>, <position>, <clear>, <delay>
:TRIGger:LOAD "LoopUntilEvent", <eventConstant>, <position>, <clear>, <delay>,
   "<readingBuffer>"
```

`<eventConstant>`	The event that ends infinite triggering or readings set to occur before the trigger; see **Details**
`<position>`	The number of readings to make in relation to the size of the reading buffer; enter as a percentage
`<clear>`	To clear previously detected trigger events when entering the wait block (default): `ENTer` To immediately act on any previously detected triggers and not clear them: `NEVer`
`<delay>`	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
`<readingBuffer>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

The event constant is the event that ends infinite triggering or ends readings set to occur before the trigger and start post-trigger readings. The trigger model makes readings until it detects the event constant. After the event, it makes a finite number of readings, based on the setting of the trigger position.

The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled. For example, if this is set to 75 for a reading buffer that holds 10,000 readings, the trigger model makes 2500 readings after it detects the source event. There will be 7500 pre-trigger readings and 2500 post-trigger readings.

The instrument makes two sets of readings. The first set is made until the trigger event occurs. The second set is made after the trigger event occurs, up to the number of readings calculated by the position parameter.

You cannot have the event constant set at none when you run this predefined trigger model.

You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

Trigger events	
Event description	**Event constant**
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	`LAN<n>`
Trigger event blender <n> (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer <n> (1 to 4) expired	`TIMer<n>`
Source limit condition occurs	`SLIMit`

Example

`*RST`
`SENS:FUNC "CURR"`
`TRIG:LOAD "LoopUntilEvent", DISP, 25`
`INIT`
	Reset the instrument.
	Set the instrument to measure DC current.
	Set the LoopUntilEvent trigger model to make measurements until the front-panel TRIGGER key is pressed after starting the trigger model, then make measurements that constitute 75 % of the reading buffer.
	Start the trigger model.

Also see

None

:TRIGger:LOAD "SimpleLoop"

This command loads a predefined trigger model configuration.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "SimpleLoop", <count>
:TRIGger:LOAD "SimpleLoop", <count>, <delay>
:TRIGger:LOAD "SimpleLoop", <count>, <delay>, "<bufferName>"
```

`<count>`	The number of measurements the instrument will make
`<delay>`	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This command sets up a loop that sets a delay, makes a measurement, and then repeats the loop the number of times you define in the count parameter.

After selecting a trigger model template, you can view the trigger model blocks in a graphical format by pressing the front-panel **MENU** key and under Trigger, selecting **Configure**. You can also add or delete blocks and change trigger model settings from this screen.

You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

This command replaces the `TRIGger:LOAD:LOOP:SIMPle` command, which is deprecated.

Example

`*RST`	Reset the instrument and set it to measure current with automatic range setting.
`SENS:FUNC "CURR"`	
`SENS:CURR:RANG:AUTO ON`	
`SOUR:FUNC VOLT`	Source 5 V with a source delay of 0.1 s.
`SOUR:DEL 0.1`	
`SOUR:VOLT 5`	
`SOUR:VOLT:ILIM 0.01`	Set a current limit of 0.01 A.
`TRIG:LOAD "SimpleLoop", 10`	Set a simple trigger loop with a count of 10.
`OUTP ON`	Turn the output on.
`INIT`	Start the trigger model.
`*WAI`	Postpone execution of subsequent commands until all previous commands are finished.
`TRAC:DATA? 1, 10, "defbuffer1", SOUR, READ, REL`	Read data and store the source, reading, and relative time.

Also see

None

:TRIGger:LOAD "SortBinning"

This command loads a predefined trigger model configuration that sets up a sorting operation.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
   <limit4High>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
   <limit4High>, <limit4Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
   <limit4High>, <limit4Low>, <limit4Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
   <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
   <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
   <limit4High>, <limit4Low>, <limit4Pattern>, "<bufferName>"
```

<components>	The number of components to measure
<startInLine>	The input line that starts the test; 5 for digital line 5, 6 for digital line 6; default is 5
<startDelay>	The delay time before each measurement (167 ns to 10 ks); default is 0 for no delay
<endDelay>	The delay time after the measurement (167 ns to 10 ks); default is 0 for no delay

<limit*x*High>	*x* is limit 1, 2, 3, or 4; the upper limit that the measurement is compared against
<limit*x*Low>	*x* is 1, 2, 3, or 4; the lower limit that the measurement is compared against
<limit1Pattern>	The bit pattern that is sent when the measurement passes limit 1; range 1 to 15; default is 1
<limit2Pattern>	The bit pattern that is sent when the measurement passes limit 2; range 1 to 15; default is 2
<limit3Pattern>	The bit pattern that is sent when the measurement passes limit 3; range 1 to 15; default is 4
<limit4Pattern>	The bit pattern that is sent when the measurement passes limit 4; range 1 to 15; default is 8
<allPattern>	The bit pattern that is sent when all limits have failed; 1 to 15; default is 15
<bufferName>	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This trigger model template allows you to sort components and place them into up to four bins, based on the comparison to limits.

To set a limit as unused, set the high value for the limit to be less than the low limit.

All limit patterns and the all fail pattern are sent on digital I/O lines 1 to 4, where 1 is the least significant bit.

After selecting a trigger model template, you can view the trigger model blocks in a graphical format by pressing the front-panel **MENU** key and under Trigger, selecting **Configure**. You can also add or delete blocks and change trigger model settings from this screen. You can use the `TRIGger:BLOCk:LIST?` command to view the trigger model blocks in a list format.

Also see

None

:TRIGger:STATe?

This command returns the present state of the trigger model.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:STATe?
```

Details

This command returns the state of the trigger model. The instrument checks the state of a started trigger model every 100 ms.

This command returns the trigger state and the block that the trigger model last executed.

The trigger model states are:

- Idle: The trigger model is stopped.
- Running: The trigger model is running.
- Waiting: The trigger model has been in the same wait block for more than 100 ms.
- Empty: The trigger model is selected, but no blocks are defined.
- Building: Blocks have been added.
- Failed: The trigger model is stopped because of an error.
- Aborting: The trigger model is stopping because of a user request.
- Aborted: The trigger model is stopped because of a user request.

Example

`:TRIG:STAT?`	An example output if the trigger model is inactive and ended at block 9 would be: `IDLE;IDLE;9`

Also see

None

:TRIGger:TIMer<n>:CLEar

This command clears the timer event detector and overrun indicator for the specified trigger timer number.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:TIMer<n>:CLEar
```

`<n>`	Trigger timer number (1 to 4)

Details

This command sets the timer event detector to the undetected state and resets the overrun indicator.

Example

`:TRIG:TIM1:CLEar`	Clears trigger timer 1.

Also see

:TRIGger:TIMer<n>:COUNt (on page 6-196)
:TRIGger:TIMer<n>:STARt:OVERrun? (on page 6-200)

:TRIGger:TIMer<n>:COUNt

This command sets the number of events to generate each time the timer generates a trigger event or is enabled as a timer or alarm.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	1

Usage

```
:TRIGger:TIMer<n>:COUNt <count>
:TRIGger:TIMer<n>:COUNt?
```

`<n>`	Trigger timer number (1 to 4)
`<count>`	The number of times to repeat the trigger (0 to 1,048,575)

Details

If *count* is set to a number greater than 1, the timer automatically starts the next trigger timer delay at the expiration of the previous delay.

Set *count* to zero (0) to cause the timer to generate trigger events indefinitely.

If you use the trigger timer with a trigger model, make sure the count value is the same or more than any count values expected in the trigger model.

Example 1

`TRIG:TIM2:COUN 4`	Set the number of events to generate for trigger timer 2 to four.

Example 2

```
*RST
TRIG:TIM4:DEL 0.5
TRIG:TIM4:STAR:STIM NOT8
TRIG:TIM4:STAR:GEN OFF
TRIG:TIM4:COUN 20
TRIG:TIM4:STAT ON

TRIG:LOAD "Empty"
TRIG:BLOC:BUFF:CLEAR 1, "defbuffer1"
TRIG:BLOC:NOT 2, 8
TRIG:BLOC:WAIT 3, TIM4
TRIG:BLOC:MEAS 4, "defbuffer1"
TRIG:BLOC:BRAN:COUN 5, 20, 3
INIT
TRAC:ACT? "defbuffer1"
```

Set trigger timer 4 to have a 0.5 s delay.
Set the stimulus for trigger timer 4 to be the notify 8 event.
Set the trigger timer 4 stimulus to off.
Set the timer event to occur when the timer delay elapses.
Set the trigger timer 4 count to 20.
Enable trigger timer 4.

Clear the trigger model.
Set trigger model block 1 to clear the buffer.
Set trigger model block 2 to generate the notify 8 event.
Set trigger model block 3 to wait for trigger timer 4 to occur.
Set trigger model block 4 to make a measurement and store it in default buffer 1.
Set trigger model block 5 to repeat the trigger model 20 times, starting at block 3.
Start the trigger model.
Output the number of entries in default buffer 1.

Output:
```
20
```

Also see

:TRIGger:TIMer<n>:CLEar (on page 6-195)
:TRIGger:TIMer<n>:DELay (on page 6-198)

:TRIGger:TIMer<n>:DELay

This command sets and reads the timer delay.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	10e-6 (10 µs)

Usage

```
:TRIGger:TIMer<n>:DELay <interval>
:TRIGger:TIMer<n>:DELay?
```

<n>	Trigger timer number (1 to 4)
<interval>	Delay interval (8e-6 s to 100,000 s)

Details

Each time the timer is triggered, it uses this delay period.

Reading this command returns the delay interval that will be used the next time the timer is triggered.

Example

```
TRIG:TIM2:DEL 50E-6
```
Set trigger timer 2 to delay for 50 µs.

Also see

None

:TRIGger:TIMer<n>:STARt:FRACtional

This command configures an alarm or a time in the future when the timer will start.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	0

Usage

```
:TRIGger:TIMer<n>:STARt:FRACtional <time>
:TRIGger:TIMer<n>:STARt:FRACtional?
```

<n>	Trigger timer number (1 to 4)
<time>	The time in fractional seconds (0 to < 1 s)

Details

This command configures the alarm of the timer.

When the timer is enabled, the timer starts immediately if the timer is configured for a start time in the past or if it is in the future.

Example

`TRIG:TIM1:STAR:SEC 60`
`TRIG:TIM1:START:FRAC 0.5`
`TRIG:TIM1:STAT ON`
	Set the timer for 60.5 s.
	Enable the trigger timer for timer 1.

Also see

:TRIGger:TIMer<n>:STARt:SEConds (on page 6-201)
:TRIGger:TIMer<n>:STATe (on page 6-203)

:TRIGger:TIMer<n>:STARt:GENerate

This command specifies when timer events are generated.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	0 (OFF)

Usage

```
:TRIGger:TIMer<n>:STARt:GENerate <state>
:TRIGger:TIMer<n>:STARt:GENerate?
```

`<n>`	Trigger timer number (1 to 4)
`<state>`	Generate a timer event when the timer delay elapses: `OFF` or `0` Generate a timer event when the timer starts and when the delay elapses: `ON` or `1`

Details

When this is set to on, a trigger event is generated immediately when the timer is triggered.

When it is set to off, a trigger event is generated when the timer elapses. This generates the event `TIMER`*N*.

Example

`TRIG:TIM3:STAR:GEN ON`	Set trigger timer 3 to generate an event when the timer starts and when the timer delay elapses.

Also see

None

:TRIGger:TIMer<n>:STARt:OVERrun?

This command indicates if an event was ignored because of the event detector state.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
:TRIGger:TIMer<n>:STARt:OVERrun?
```

<n>	Trigger timer number (1 to 4)

Details

This command indicates if an event was ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the timer itself. It does not indicate if an overrun occurred in any other part of the trigger model or in any other construct that is monitoring the delay completion event. It also is not an indication of a delay overrun.

This returns `0` if there is no overrun or `1` if there is an overrun.

Example

`TRIG:TIM1:STAR:OVER?`	Checks the overrun status on trigger timer 1.

Also see

None

:TRIGger:TIMer<n>:STARt:SEConds

This command configures an alarm or a time in the future when the timer will start.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	0

Usage

```
:TRIGger:TIMer<n>:STARt:SEConds <time>
:TRIGger:TIMer<n>:STARt:SEConds?
```

<n>	Trigger timer number (1 to 4)
<time>	The time: 0 to 2,147,483,647 s

Details

This command configures the alarm of the timer.

When the timer is enabled, the timer starts immediately if the timer is configured for a start time that has passed.

Example

```
TRIG:TIM1:STAR:SEC 60
TRIG:TIM1:START:FRAC 0.5
TRIG:TIM1:STAT ON
```

Set the timer for 60.5 s.
Enable the trigger timer for timer 1.

Also see

:TRIGger:TIMer<n>:STATe (on page 6-203)

:TRIGger:TIMer<n>:STARt:STIMulus

This command describes the event that starts the trigger timer.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	NONE

Usage

```
:TRIGger:TIMer<n>:STARt:STIMulus <event>
:TRIGger:TIMer<n>:STARt:STIMulus?
```

<n>	Trigger timer number (1 to 4)
<event>	The event that starts the trigger timer

Details

Set this attribute any trigger event to start the timer when that event occurs.

Set this attribute to zero (0) to disable event processing and use the timer as a timer or alarm based on the start time.

Trigger events are described in the table below.

Trigger events	
Event description	**Event constant**
No trigger event	NONE
Front-panel TRIGGER key press	DISPlay
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	NOTify<n>
A command interface trigger: • Any remote interface: *TRG • GPIB only: GET bus command • VXI-11: VXI-11 command device_trigger	COMMand
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	DIGio<n>
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	TSPLink<n>
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	LAN<n>
Trigger event blender <n> (1 or 2), which combines trigger events	BLENder<n>
Trigger timer <n> (1 to 4) expired	TIMer<n>
Source limit condition occurs	SLIMit

Example

`*RST`
`DIG:LINE1:MODE TRIG,IN`
`DIG:LINE2:MODE TRIG,OUT`
`TRIG:TIM1:DEL 35e-3`
`TRIG:TIM1:STAR:STIM DIG1`
`TRIG:DIG2:OUT:STIM TIM1`
	Reset the instrument to default settings. Set digital I/O line 1 for use as a trigger input. Set digital I/O line 2 for use as a trigger output.
	Set timer 1 to delay 35 ms.
	Set timer 1 to start delaying once the digital I/O 1 event is detected.
	Set digital I/O line 2 to output a pulse once the timer 1 event is detected.

Also see

None

:TRIGger:TIMer<n>:STATe

This command enables the trigger timer.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	0 (OFF)

Usage

```
:TRIGger:TIMer<n>:STATe <state>
:TRIGger:TIMer<n>:STATe?
```

`<n>`	Trigger timer number (1 to 4)
`<state>`	Disable the trigger timer: `OFF` or `0` Enable the trigger timer: `ON` or `1`

Details

When this command is set to on, the timer performs the delay operation.

When this command is set to off, there is no timer on the delay operation.

You must enable a timer before it can use the delay settings or the alarm configuration. For expected results from the timer, it is best to disable the timer before changing a timer setting, such as delay or start seconds.

To use the timer as a simple delay or pulse generator with digital I/O lines, make sure the timer start time in seconds and fractional seconds is configured for a time in the past. To use the timer as an alarm, configure the timer start time in seconds and fractional seconds for the desired alarm time.

Example 1

`DIG:LINE3:MODE TRIG,OUT`
`TRIG:DIG3:OUT:STIM TIM2`
`SYSTem:TIME?`
`TRIG:TIM2:START:SECONDS <current time> + 60`
`TRIG:TIM2:STAT ON`
	To configure timer 2 for an alarm to fire 1 minute from now and output a pulse on digital I/O line 3, query to get the current time. Add 60 s to that value and use that to configure the start seconds. Enable the timer.

Example 2

```
*RST
DIG:LINE5:MODE TRIG,OUT
TRIG:DIG5:OUT:STIM TIM3
TRIG:TIM3:DEL 3e-3
TRIG:TIM3:COUNT 5
TRIG:TIM3:STAT ON
```

Configure timer 3 to generate 5 pulses on digital I/O line 5 that are 3 ms apart.

Example 3

```
*RST
DIG:LINE3:MODE TRIG,IN
DIG:LINE5:MODE TRIG,OUT
TRIG:DIG5:OUT:STIM TIM3
TRIG:TIM3:DEL 3e-3
TRIG:TIM3:COUNT 5
TRIG:TIM3:START:STIM DIG3
TRIG:TIM3:STAT ON
```

Configure timer 3 to generate 5 pulses on digital I/O line 5 that are 3 ms apart when a digital input is detected on digital line 3.

Also see

None