Table of contents

Section 1

TSP command reference

In this section:

TSP commands

The TSP commands available for the instrument are listed in alphabetical order.

bufferVar.measurefunctions

This attribute contains the measurement function that was used to acquire a reading stored in a specified reading buffer.Usage

```
measurefunction = bufferVar.measurefunctions[N]
```

measurefunction	The measurement function used (`Current`, `Voltage`, `Ohms`, or `Watts`) to acquire reading number *N* in the specified buffer
bufferVar	The reading buffer; can be a dynamically allocated buffer (user-defined), or a dedicated reading buffer (such as `smua.nvbuffer1`)
N	The reading number (`1` to *`bufferVar`*`.n`)

Details

The `measurefunctions` buffer recall attribute is like an array (a Lua table) of strings indicating the function measured for the reading.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example 1

```
measurefunction = smua.nvbuffer1.measurefunctions[5]
```

Store the measure function used to make reading number 5.

Example 2

```
printbuffer(1, 5, smua.nvbuffer1.measurefunctions)
```

Print the measurement function that was used to measure the first five readings saved in dedicated reading buffer 1.

Example output:

```
Current, Current, Current, Current, Current
```

Also see

bufferVar.measureranges (on page 1-3)
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.sourcefunctions (on page 1-5)
bufferVar.sourceoutputstates (on page 1-6)
bufferVar.sourceranges (on page 1-7)
bufferVar.sourcevalues (on page 1-8)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.measureranges

This attribute contains the measurement range values that were used for readings stored in a specified buffer.

Usage

```
measurerange = bufferVar.measureranges[N]
```

measurerange	The measurement range used to acquire reading number *N* in the specified buffer
bufferVar	The reading buffer; can be a dynamically allocated buffer (user-defined), or a dedicated reading buffer (such as `smua.nvbuffer1`)
N	The reading number (1 to *bufferVar*.n)

Details

The `measureranges` buffer recall attribute is like an array (a Lua table) of full-scale range values for the measure range used when the measurement was made.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example 1

```
measurerange = smua.nvbuffer1.measureranges[1]
```

Store the measure range that was used to make reading number 1.

Example 2

```
printbuffer(1, 10, smua.nvbuffer1.measureranges)
```

Print the range values that were used for the first 10 readings saved in dedicated reading buffer 1.
Example output:

```
1.00000e-07, 1.00000e-07,
1.00000e-07, 1.00000e-07,
1.00000e-07, 1.00000e-07,
1.00000e-07, 1.00000e-07,
1.00000e-07, 1.00000e-07
```

Also see

bufferVar.measurefunctions (on page 1-2)
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.sourcefunctions (on page 1-5)
bufferVar.sourceoutputstates (on page 1-6)
bufferVar.sourceranges (on page 1-7)
bufferVar.sourcevalues (on page 1-8)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.n

This attribute contains the number of readings in the buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

```
numberOfReadings = bufferVar.n
```

numberOfReadings	The number of readings stored in the buffer
bufferVar	The reading buffer

Details

This read-only attribute contains the number of readings presently stored in the buffer.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example

```
numberOfReadings = smua.nvbuffer1.n
print(numberOfReadings)
```

Reads the number of readings stored in dedicated reading buffer 1 (source-measure unit (SMU) channel A).
Output:
```
1.25000+02
```
The above output indicates that there are 125 readings stored in the buffer.

Also see

bufferVar.measurefunctions (on page 1-2)
bufferVar.measureranges (on page 1-3)
bufferVar.readings (on page 1-5)
bufferVar.sourcefunctions (on page 1-5)
bufferVar.sourceoutputstates (on page 1-6)
bufferVar.sourceranges (on page 1-7)
bufferVar.sourcevalues (on page 1-8)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.readings

This attribute contains the readings stored in a specified reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

`reading` `=` *`bufferVar`*`.readings[`*`N`*`]`

`reading`	The value of the reading in the specified reading buffer
`bufferVar`	The reading buffer
`N`	The reading number *`N`*; can be any value from 1 to the number of readings in the buffer; use the *`bufferVar.`*`n` command to determine the number of readings in the buffer

Details

The *`bufferVar`*`.readings` buffer recall attribute is like an array (a Lua table) of the readings stored in the reading buffer. This array holds the same data that is returned when the reading buffer is accessed directly; that is, `rb[2]` and `rb.readings[2]` access the same value.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Also see

bufferVar.sourcefunctions

This attribute contains the source function that was being used when the readings were stored in a specified reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

`sourcefunction` `=` *`bufferVar`*`.sourcefunctions[`*`N`*`]`

`sourcefunction`	The source function used (`Current` or `Voltage`) to acquire reading number *`N`* in the specified buffer
`bufferVar`	The reading buffer; can be a dynamically allocated buffer (user-defined), or a dedicated reading buffer (such as `smua.nvbuffer1`)
`N`	The reading number (`1` to *`bufferVar`*`.n`)

Details

The *`bufferVar`*`.sourcefunctions` buffer recall attribute is like an array (a Lua table) of strings indicating the source function at the time of the measurement.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example 1

```
sourcefunction = smua.nvbuffer1.sourcefunctions[3]
print(sourcefunction)
```

Store the source function used to make reading number 3 and output the value.

Example 2

```
printbuffer(1, 10, smua.nvbuffer1.sourcefunctions)
```

Print the source function used for 10 readings stored in dedicated reading buffer 1.
Example output:

```
Voltage, Voltage, Voltage, Voltage, Voltage, Voltage, Voltage, Voltage, Voltage, Voltage
```

Also see

bufferVar.measurefunctions (on page 1-2)
bufferVar.measureranges (on page 1-3)
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.sourceoutputstates (on page 1-6)
bufferVar.sourceranges (on page 1-7)
bufferVar.sourcevalues (on page 1-8)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.sourceoutputstates

This attribute indicates the state of the source output for readings that are stored in a specified buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

`state = bufferVar.sourceoutputstates[N]`

`state`	The output state (`Off` or `On`) when reading *`N`* of the specified buffer was acquired
`bufferVar`	The reading buffer; can be a dynamically allocated buffer (user-defined), or a dedicated reading buffer (such as `smua.nvbuffer1`)
`N`	The reading number (1 to *`bufferVar`*`.n`)

Details

The *`bufferVar`*`.sourceoutputstates` buffer recall attribute is similar to an array (a Lua table) of strings. This array indicates the state of the source output (`Off` or `On`) at the time of the measurement.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example

```
printbuffer(1, 1, smua.nvbuffer1.sourceoutputstates)
```

Print the source output for the first reading stored in dedicated reading buffer 1.
Example output:
```
On
```

Also see

bufferVar.measurefunctions (on page 1-2)
bufferVar.measureranges (on page 1-3)
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.sourcefunctions (on page 1-5)
bufferVar.sourceranges (on page 1-7)
bufferVar.sourcevalues (on page 1-8)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.sourceranges

This attribute contains the source range that was used for readings stored in a specified reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

sourcerange = *bufferVar*.sourceranges[*N*]

sourcerange	The source range used to acquire reading number *N* in the specified buffer
bufferVar	The reading buffer; can be a dynamically allocated buffer (user-defined), or a dedicated reading buffer (such as `smua.nvbuffer1`)
N	The reading number (1 to *bufferVar*.n)

Details

The *bufferVar*`.sourceranges` buffer recall attribute is like an array (a Lua table) of full-scale range values for the source range used when the measurement was made.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example 1

```
sourcerange = smua.nvbuffer1.sourceranges[1]
```

Store the source range that was used for the first reading stored in dedicated reading buffer 1.

Example 2

```
printbuffer(1, 6, smua.nvbuffer1.sourceranges)
```

Print the source ranges that were used for the first 6 readings stored in source-measure unit (SMU) A, buffer 1.
Example output:

```
1.00000e-04, 1.00000e-04, 1.00000e-04, 1.00000e-04, 1.00000e-04, 1.00000e-04
```

Also see

bufferVar.measurefunctions (on page 1-2)
bufferVar.measureranges (on page 1-3)
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.sourcefunctions (on page 1-5)
bufferVar.sourceoutputstates (on page 1-6)
bufferVar.sourcevalues (on page 1-8)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.sourcevalues

When enabled by the *bufferVar*`.collectsourcevalues` attribute, this attribute contains the source levels being output when readings in the reading buffer were acquired.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

`sourcevalue = bufferVar.sourcevalues[N]`

sourcevalue	The output value of the source when reading *N* of the specified buffer was acquired
bufferVar	The reading buffer; can be a dynamically allocated buffer (user-defined) or a dedicated reading buffer (such as `smua.nvbuffer1`)
N	The reading number (1 to *bufferVar*.n)

Details

If the *bufferVar*`.collectsourcevalues` attribute is enabled before readings are made, the *bufferVar*`.sourcevalues` buffer recall attribute is like an array (a Lua table) of the sourced value in effect at the time of the reading.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example 1

`sourcevalue = smua.nvbuffer1.sourcevalues[1]`	Get the sourced value of the first reading stored in dedicated reading buffer 1.

Example 2

`printbuffer(1, 6, smua.nvbuffer1.sourcevalues)`	Print the sourced value of the first 6 readings stored in source-measure unit (SMU) A, buffer 1.
	Example output:
	`1.00000e-04, 1.00000e-04,`
	`1.00000e-04, 1.00000e-04,`
	`1.00000e-04, 1.00000e-04`

Also see

bufferVar.collectsourcevalues
bufferVar.measurefunctions (on page 1-2)
bufferVar.measureranges (on page 1-3)
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.sourcefunctions (on page 1-5)
bufferVar.sourceoutputstates (on page 1-6)
bufferVar.sourceranges (on page 1-7)
bufferVar.statuses (on page 1-10)
bufferVar.timestamps (on page 1-12)
Reading buffers

bufferVar.statuses

This attribute contains the status values of readings in the reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

statusInformation = *bufferVar*.statuses[*N*]

statusInformation	The status value when reading *N* of the specified buffer was acquired
bufferVar	The reading buffer
N	The reading number *N*; can be any value from 1 to the number of readings in the buffer; use the *bufferVar*.n command to determine the number of readings in the buffer

Details

This read-only buffer recall attribute is like an array (a Lua table) of the status values for all the readings in the buffer. The status values are floating-point numbers that encode the status value; see the following table for values.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Buffer status bits

Bit	Name	Hex	Description
B1	Overtemp	0x02	Over temperature condition
B2	AutoRangeMeas	0x04	Measure range was autoranged
B3	AutoRangeSrc	0x08	Source range was autoranged
B4	4Wire	0x10	4-wire (remote) sense mode enabled
B5	Rel	0x20	Relative offset applied to reading
B6	Compliance	0x40	Source function was limited because the complementary function would be over the compliance limit
B7	Filtered	0x80	Reading was filtered

Example

`reset()`	Reset the instrument.
`smua.source.func = smua.OUTPUT_DCVOLTS`	Set the voltage source function to DC volts.
`smua.source.autorangev = smua.AUTORANGE_ON`	Set the range to auto.
`smua.source.levelv = 5`	Set the voltage source to 5 V.
`smua.source.limiti = 10e-3`	Set current measure limit to 10 mA.
`smua.measure.rangei = 10e-3`	Set the current measure range to 10 mA.
`smua.source.output = smua.OUTPUT_ON`	Turn on the output.
`print(smua.measure.i(smua.nvbuffer1))`	Print and place the current reading in the reading buffer.
`smua.source.output = smua.OUTPUT_OFF`	Turn off the output.

`print(smua.nvbuffer1.statuses[1])`	Output status value of the first measurement in the reading buffer.
	Output example:
	`3.99470e-06`
	`4.00000e+00`

Also see

bufferVar.timestamps (on page 1-12)

Reading buffers

bufferVar.measurefunctions (on page 1-2)

bufferVar.measureranges (on page 1-3)

bufferVar.n (on page 1-4)

bufferVar.readings (on page 1-5)

bufferVar.sourcefunctions (on page 1-5)

bufferVar.sourceoutputstates (on page 1-6)

bufferVar.sourceranges (on page 1-7)

bufferVar.sourcevalues (on page 1-8)

bufferVar.timestamps

When enabled by the *bufferVar*`.collecttimestamps` attribute, this attribute contains the timestamp when each reading saved in the specified reading buffer occurred.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Clearing the buffer	See **Details**	Not applicable

Usage

```
timestamp = bufferVar.timestamps[N]
```

timestamp	The complete timestamp (including date, time, and fractional seconds) of reading number *N* in the specified reading buffer when the reading was acquired
bufferVar	The reading buffer ; can be a dynamically allocated user-defined buffer or a dedicated reading buffer
N	The reading number (1 to *bufferVar*`.n`)

Details

The *bufferVar*`.timestamps` information from a reading buffer is only available if the *bufferVar*`.collecttimestamps` attribute is set to 1 (default setting). If it is set to 0, you cannot access any time information from a reading buffer.

If enabled, this buffer recall attribute is like an array (a Lua table) that contains timestamps, in seconds, of when each reading occurred. These are relative to the *bufferVar*`.basetimestamp` for the buffer.

For dedicated reading buffers, all buffer attributes are saved to nonvolatile memory only when the reading buffer is saved to nonvolatile memory.

Example

`timestamp = smua.nvbuffer1.timestamps[1]`	Get the timestamp of the first reading stored in source-measure unit (SMU) A, buffer 1.

Also see

bufferVar.clear()
bufferVar.collecttimestamps
bufferVar.n (on page 1-4)
bufferVar.readings (on page 1-5)
bufferVar.statuses (on page 1-10)
Reading buffers

digio.readbit()

This function reads one digital I/O line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
data = digio.readbit(N)
```

value	The value to read, 1 or 0
N	Digital I/O line number to be read: 1 to 18

Details

A returned value of zero (0) indicates that the line is low. A returned value of one (1) indicates that the line is high.

Example

```
-Read the value of digital I/O line 2
local value = digio.readbit(2)
print("The value of line 2 is: " .. value)
```

Reads the state of digital I/O line 2.
Output

```
2.00<expNote+00
```

Also see

digio.readport() (on page 1-14)
digio.writebit() (on page 1-29)
digio.writeport() (on page 1-30)
Digital I/O port ***add link later***

digio.readport()

This function reads the digital I/O port

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
data = digio.readport()
```

data	The present value of the input lines on the digital I/O port

Details

The binary equivalent of the returned value indicates the value of the input lines on the I/O port. The least significant bit (bit B1) of the binary number corresponds to line 1; bit B14 corresponds to line 14.

For example, a returned value of 170 has a binary equivalent of 000000010101010, which indicates that lines 2, 4, 6, and 8 are high (1), and the other 10 lines are low (0).

Example

```
data = digio.readport()
print(data)
```

Assume lines 2, 4, 6, and 8 are set high when the I/O port is read.
Output:
1.7000000e+02
This is binary 10101010.

Also see

digio.readbit() (on page 1-13)
digio.writebit() (on page 1-29)
digio.writeport() (on page 1-30)
Digital I/O port ***add link later

digio.trigger[N].assert()

This function asserts a trigger pulse on one of the digital I/O lines

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
digio.trigger[N].assert()
```

N	Digital I/O trigger line (1 to 18)

Details

This command asserts a trigger on the specified trigger line. When a trigger is asserted, it generates a pulse with the duration specified by `digio.trigger[N].pulsewidth`.

Examples

```
digio.trigger[2].assert()
```

Asserts digital I/O trigger line 2.

```
digio.trigger[2].logic = digio.LOGIC_NEGATIVE
digio.trigger[2].mode = digio.MODE_TRIGGER_OUT
digio.trigger[2].pulsewidth = 0
digio.trigger[2].assert()
digio.trigger[2].release()
```

Negative trigger out. This example uses indefinite-length triggers, as it sets the pulse width to 0 and uses `digio.trigger[N].release()` rather than using automatic pulse generation.

```
digio.trigger[2].logic = digio.LOGIC_POSITIVE
digio.trigger[2].mode = digio.MODE_TRIGGER_OUT
digio.trigger[2].pulsewidth = 10e-6
digio.trigger[2].assert()}}
```

Positive trigger out.

Also see

digio.trigger[N].mode (on page 1-20)
digio.trigger[N].pulsewidth (on page 1-24)

digio.trigger[N].clear()

This function clears the trigger event on a digital I/O line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
N = digio.trigger[N].clear()
```

N	Digital I/O trigger line (1 to 18)

Details

The event detector of a trigger enters the detected state when an event is detected. It is cleared when `digio.trigger[N].wait()` or `digio.trigger[N].clear()` is called.

`digio.trigger[N].clear()` clears the event detector of the specified trigger line, discards the history of the trigger line, and clears the `digio.trigger[N].overrun` attribute.

Example

`digio.trigger[2].clear()`
Clears the trigger event on digital I/O line 2.

Also see

digio.trigger[N].overrun (on page 1-23)
digio.trigger[N].wait (on page 1-28)

digio.trigger[N].edge

This command configures the edge detection mode for a digital I/O line.

Type	TSP-Link accssible	Affected by	Where saved	Default value
Attribute (RW)	Yes	digio.trigger[N].reset Instrument reset Power cycle	Not applicable	digio.EDGE_FALLING

Usage

```
triggerEdge = digio.trigger[N].edge
digio.trigger[N].edge = triggerEdge
```

triggerEdge	The trigger edge selection; see **Trigger edge values** for values
N	Digital I/O trigger line (1 to 18)

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.

This command is used for input trigger detection. To control output trigger generation, use the `digio.trigger[N].logic` command.

Set *triggerEdge* to one of the following values:

Trigger edge values

triggerEdge	**Description**
digio.EDGE_FALLING	Detects falling-edge triggers as input when the line is configured as an input or open drain
digio.EDGE_RISING	Detects rising-edge triggers as input when the line is configured as an open drain
digio.EDGE_EITHER	Detects rising- or falling-edge triggers as input when the line is configured as an input or open drain

Example

```
digio.trigger[3].edge = digio.EDGE_RISING
```

Sets the trigger edge detection mode for I/O line 3 to `digio.EDGE_RISING`. This will configure line 3 to detect rising edges as input triggers.

Also see

digio.trigger[N].clear() (on page 1-16)
digio.trigger[N].logic (on page 1-19)
digio.trigger[N].mode (on page 1-20)
digio.trigger[N].reset (on page 1-25)
digio.writeport() (on page 1-30)

digio.trigger[N].EVENT_ID

This constant identifies the trigger event generated by the digital I/O line *N*.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Constant	Yes			

Usage

`eventID = digio.trigger[N].EVENT_ID`

eventID	The trigger event number
N	Digital I/O trigger line (1 to 18)

Details

To have another trigger object respond to trigger events generated by the trigger line, set the stimulus attribute of the other object to the value of this constant.

Example

1

```
digio.trigger[5].stimulus = digio.trigger[3].EVENT_ID
```

Uses a trigger event on digital I/O trigger line 3 to be the stimulus for digital I/O trigger line 5.

Also see

None

digio.trigger[N].logic

This attribute sets the output logic of the trigger event generator to positive or negative for the specified line.

Type	TSP-Link accssible	Affected by	Where saved	Default value
Attribute (RW)	Yes	digio.trigger[N].reset Instrument reset Power cycle	Not applicable	digio.LOGIC_NEGATIVE

Usage

```
triggerLogic = digio.trigger[N].logic
digio.trigger[N].logic = triggerLogic
```

triggerLogic	The trigger mode; see **Details** for values
N	Digital I/O trigger line (1 to 18)

Details

This attribute is used for output trigger generation. To control input trigger detection, use the `digio.trigger[N].edge` command.

Set *triggerLogic* to one of the following values:

Trigger logic level

triggerLogic	**Description**
`digio.LOGIC_POSITIVE`	Asserts a TTL-high pulse for output
`digio.LOGIC_NEGATIVE`	Assert a TTL-low pulse for output

Example

```
digio.trigger[3].logic = digio.LOGIC_POSITIVE
```

Sets I/O line 3 mode to be a trigger output and sets the output logic of the trigger event generator to negative.

Also see

digio.trigger[N].edge (on page 1-17)
digio.trigger[N].mode (on page 1-20)
digio.trigger[N].reset (on page 1-25)

digio.trigger[N].mode

This attribute 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.

Usage

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	digio.trigger[N].reset() Instrument reset Power cycle	Not applicable	digio.MODE_DIGITAL_IN

Usage

```
triggerMode = digio.trigger[N].mode
digio.trigger[N].mode = triggerMode
```

triggerMode	The trigger mode; see **Details** for values
N	Digital I/O trigger line (1 to 18)

Details

Set *triggerMode* to one of the following values:

Trigger mode values

triggerMode	**Description**
`digio.MODE_DIGITAL_IN`	Digital control, input
`digio.MODE_DIGITAL_OUT`	Digital control, output
`digio.MODE_DIGITAL_OPEN_DRAIN`	Digital control, open drain
`digio.MODE_TRIGGER_IN`	Trigger control, input
`digio.MODE_TRIGGER_OUT`	Trigger control, output
`digio.MODE_TRIGGER_OPEN_DRAIN`	Trigger control, open drain
`digio.MODE_SYNCHRONOUS_MASTER`	Synchronous master
`digio.MODE_SYNCHRONOUS_ACCEPTOR`	Synchronous acceptor

You can use this command to place 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 synchronous 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.

The following settings of *triggerMode* set the line for direct control as a digital line:

- `digio.MODE_DIGITAL_IN`: The instrument automatically detects externally generated logic levels. You can read an input line, but you cannot write to it.
- `digio.MODE_DIGITAL_OUT`: 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.
- `digio.MODE_DIGITAL_OPEN_DRAIN`: Configures the line to be an open-drain signal. The line can serve as an input, an output or both. When a digital I/O line is used as an input in open-drain mode, you must write a `1` to it.

The following settings of *triggerMode* set the line for direct control as a digital line:

- `digio.MODE_TRIGGER_IN`: The line automatically responds to and detects externally generated triggers. It detects falling-edge, rising-edge, or either-edge triggers as input. This line state uses the edge setting specified by the `digio.trigger.digin[N].edge` attribute.
- `digio.MODE_TRIGGER_OUT`: The line 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.
- `digio.MODE_TRIGGER_OPEN_DRAIN`: Configures the line to be an open-drain signal. You can use the line to detect input triggers or generate output triggers. This line state uses the edge setting specified by the `digio.trigger.digin[N].edge` attribute.

Example

`digio.trigger[3].mode = digio.MODE_DIGITAL_IN`
Sets the trigger mode for I/O line 3 to `digio.MODE_DIGITAL_IN`.

Also see

None

digio.trigger[N].overrun

This function returns the event detector overrun status.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	digio.trigger[N].clear() digio.trigger[N].reset()	Not applicable	Not applicable

Usage

```
overrun = digio.trigger[N].overrun
```

overrun	Trigger overrun state: true or false
N	Digital I/O trigger line (1 to 18)

Details

If this is true, 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

```
overrun = digio.trigger[1].overrun
if overrun then
     print("Trigger overrun set on digital input 1")
else
     print("No trigger overrun on digital input 1")
end
```

Prints the status of the event detector overrun.

Also see

digio.trigger[N].clear() (on page 1-16)
digio.trigger[N].reset() (on page 1-25)

digio.trigger[N].pulsewidth

This attribute describes the length of time that the trigger line is asserted for output triggers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	digio.trigger[N].reset Instrument reset Power cycle	Not applicable	10e-6 (10 us)

Usage

```
triggerPulseWidth = digio.trigger[N].pulsewidth
digio.trigger[N].pulsewidth = triggerPulseWidth
```

triggerPulseWidth	Pulse width, from 0.1 µs to 6.5535 us
N	Digital I/O trigger line (1 to 18)

Details

Setting the pulse width to zero (0) seconds asserts the trigger indefinitely. To release the trigger line use `digio.trigger[N].release()`.

Example

`digio.trigger[2].pulsewidth = 0.0001`
Sets the pulse width for I/O trigger line 2 to 100 us.

Also see

digio.trigger[N].assert() (on page 1-14)
digio.trigger[N].release() (on page 1-24)
digio.trigger[N].reset() (on page 1-25)

digio.trigger[N].release()

This function releases an indefinite length or latched trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function (RW)	Yes			

Usage

```
digio.trigger[N].release()
```

N	Digital I/O trigger line (1 to 18)

Details

Releases a trigger that was asserted with an indefinite pulsewidth time. It also releases a trigger that was latched in response to receiving a synchronous mode trigger. Only the specified trigger line is affected.

Examples

`digio.trigger[2].release()`
Releases digital I/O trigger line 2.

Also see

digio.trigger[N].assert() (on page 1-14)
digio.trigger[N].pulsewidth (on page 1-24)

digio.trigger[N].reset()

This function resets digital I/O line values to their factory defaults.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
digio.trigger[N].reset()
```

N	Digital I/O trigger line (1 to 18)

Details

This function resets the following attributes to their default values:

- `digio.trigger[N].mode`
- `digio.trigger[N].edge`
- `digio.trigger[N].logic`
- `digio.trigger[N].pulsewidth`
- `digio.trigger[N].stimulus`

It also clears `digio.trigger[N].overrun`.

Examples

```
digio.trigger[2].reset()
```

Resets digital I/O trigger line 2.

```
digio.trigger[2].logic = digio.LOGIC_POSITIVE
digio.trigger[2].mode = digio.MODE_TRIGGER_OUT
digio.trigger[2].pulsewidth = 0
digio.trigger[2].assert()
digio.trigger[2].release()
print("logic:",digio.trigger[2].logic, " edge:",digio.trigger[2].edge, "
   pulsewidth:", digio.trigger[2].pulsewidth, " mode:", digio.trigger[2].mode)
```

Trigger setting before and after reset.

Also see

digio.trigger[N].edge (on page 1-17)
digio.trigger[N].logic (on page 1-19)
digio.trigger[N].mode (on page 1-20)
digio.trigger[N].overrun (on page 1-23)
digio.trigger[N].pulsewidth (on page 1-24)
digio.trigger[N].stimulus (on page 1-27)

digio.trigger[N].stimulus

This attribute selects the event that causes a trigger to be asserted on the digital output line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Digital I/O trigger *N* reset.	Configuration script	digio.EVENT_NONE

Usage

```
event = digio.trigger[N].stimulus
digio.trigger[N].stimulus = event
```

event	The event to use as a stimulus; see **Details**
N	Digital I/O trigger line (1 to 18)

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**
Trigger event blender *N* (1 to 2), which combines trigger events	`digio.EVENT_BLENDERN`
A command interface trigger: ■ Any remote interface: `*TRG` ■ GPIB only: GET bus command ■ USB only: A USBTMC TRIGGER message ■ VXI-11: VXI-11 command `device_trigger`	`digio.EVENT_COMMAND`
Digital input line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line *N* (1 to 18)	`digio.EVENT_DIGION`
Front-panel TRIGGER key press	`digio.EVENT_DISPLAY`
Appropriate LXI trigger packet is received on LAN trigger object *N* (1 to 8)	`digio.EVENT_LANN`
No trigger event	`digio.EVENT_NONE`
Notify trigger block *N* (1 to 8) generates a trigger event when the trigger model executes it	`digio.EVENT_NOTIFYN`
Trigger timer *N* (1 to 4) expired	`digio.EVENT_TIMERN`
Line edge detected on TSP-Link synchronization line *N* (1 to 3)	`digio.EVENT_TSPLINKN`

Example

```
digio.trigger[2].mode = digio.MODE_TRIGGER_OUT
trigger.digout[2].stimulus = digio.EVENT_TIMER4
```

Sets the stimulus for digital I/O line 2 to be the expiration of trigger timer 4.

Also see

digio.trigger[N].assert (on page 1-14)
digio.trigger[N].clear (on page 1-16)
digio.trigger[N].mode (on page 1-20)
digio.trigger[N].reset (on page 1-25)

digio.trigger[N].wait()

This function waits for an input trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

triggered = digio.trigger[*N*].wait(*timeout*)

triggered	Trigger detected: `true` No triggers detected during the timeout period: `false`
N	Digital I/O trigger line (1 to 18)
timeout	Timeout in seconds

Details

This function pauses for up to *timeout* seconds for an input trigger. If one or more trigger events are detected since the last time `digio.trigger[N].wait()` or `digio.trigger[N].clear()` was called, this function returns a value immediately. After waiting for a trigger with this function, the event detector is automatically reset and is ready to detect the next trigger. This is true regardless of the number of events detected.

Example

```
triggered = digio.trigger[1]wait(5)
if triggered then
     print("Trigger event detected on pin 1")
else
     print("Timeout waiting for trigger event on pin 1")
end
```

Waits for up to 5 s for a trigger to be detected on trigger line 1, then outputs the results.
Output if no trigger is detected:
`Timeout waiting for trigger event on pin 1`
Output if a trigger is detected:
`Trigger event detected on pin 1`

Also see

digio.trigger[N].clear (on page 1-16)
digio.trigger[N].mode (on page 1-20)

digio.writebit()

This function writes a value to a specific digital I/O line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`digio.writebit(N, value)`

N	Digital I/O trigger line (1 to 18)
value	The value to write to the bit: ■ 0 (low) ■ Non-zero (high)

Details

If the output line is write-protected using the `digio.writeprotect` attribute, the command is ignored.

The `reset()` function does not affect the present state of the digital I/O lines.

Use the `digio.writebit()` and `digio.writeport()` commands to control the output state of the synchronization line when trigger operation is set to `digio.TRIG_BYPASS`.

The data must be zero (0) to clear the bit. Any value other than zero (0) sets the bit.

Example

`digio.writebit(2, 0)`
Sets digital I/O line 2 to low (0).

Also see

digio.readbit() (on page 1-13)
digio.readport() (on page 1-14)
digio.trigger[N].mode (on page 1-20)
digio.writeport() (on page 1-30)
digio.writeprotect (on page 1-31)

digio.writeport()

This function writes to all digital I/O lines.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
digio.writeport(data)
```

data	Value to write to the port (0 to 262143)

Details

The binary representation of *data* indicates the output pattern to be written to the I/O port. For example, a *data* value of 170 has a binary equivalent of 00000010101010. Lines 2, 4, 6, and 8 are set high (1), and the other 10 lines are set low (0).

Write-protected lines are not changed.

The `reset()` function does not affect the present states of the digital I/O lines.

Use the `digio.writebit()` and `digio.writeport()` commands to control the output state of the synchronization line when trigger operation is set to `digio.TRIG_BYPASS`.

Example

`digio.writeport(255)`
Sets digital I/O lines 1 through 8 high (binary 00000011111111)

Also see

digio.readbit() (on page 1-13)
digio.readport() (on page 1-14)
digio.writebit() (on page 1-29)
digio.writeprotect (on page 1-31)

digio.writeprotect

This attribute contains the write-protect mask that protects bits from changes from the `digio.writebit()` and `digio.writeport()` functions

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Instrument reset Power cycle	Not applicable	0

Usage

```
digio.writeprotect = writeprotectMask
writeprotectMask = digio.writeprotect
```

writeprotectMask	Sets the value that specifies the bit pattern for write-protect. Values can range from 0 to 262143.

Details

Bits that are set to one cause the corresponding line to be write-protected.

The binary equivalent of *mask* indicates the mask to be set for the I/O port. For example, a mask value of 7 has a binary equivalent of 00000000000111. This mask write-protects lines 1, 2, and 3.

Example

```
local writeProtectMask = 15
```

Protects digital I/O lines 1, 2, 3, and 4 (binary 00001111).

Also see

digio.writebit() (on page 1-29)
digio.writeport() (on page 1-30)

eventlog.clear()

This function clears the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.clear()
```

Details

This command regmoves all events from the command log.

Example

`eventlog.clear()`	
Clears the event log.	

Also see

None

eventlog.count

This attribute stores the number of events in the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes			

Usage

```
count = eventlog.count
```

count	The total number of events in the event log

Details

This attribute stores the total number of events in the event log. To get a count of errors or events of a specific type, use `eventlog.getcount()` (on page 1-39).

Example

```
count = eventlog.count
print(count)
```

Displays the `eventlog` count.

Also see

eventlog.getcount() (on page 1-39)

eventlog.disable.add()

This function will add a list of errors to the disabled list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.disable.add(list)
```

list	This can be a string in list notation or a table

Details

This command adds the error and event numbers in the list to the `eventlog.disable.list` (on page 1-34) list.

Example

Also see

eventlog.disable.list (on page 1-34)

eventlog.disable.delete()

This function will remove a list of errors from the disabled list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.disable.delete(list)
```

list	This can be a string in list notation or a table

Details

This command removes the error and event numbers in the list from the `eventlog.disable.list` (on page 1-34) list.

Example

Also see

eventlog.disable.list (on page 1-34)

eventlog.disable.list

This attribute stores the list of disabled error and event numbers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes			

Usage

```
list = eventlog.disable.list
eventlog.disable.list = list
```

list	You can set this attribute as a string in list notation or a table. When the attribute is read, it will be a string in list notation

Details

This attribute stores the list of disabled error and event numbers. Errors and events in this list will not be logged when encountered. Setting this attribute also affects the `eventlog.enable.list` (on page 1-38) attribute.

Example

`print(evenlog.disable.list)`
Prints attribute list.

Also see

eventlog.disable.add() (on page 1-33)
eventlog.disable.delete() (on page 1-33)
eventlog.disable.severity() (on page 1-35)
eventlog.disable.type() (on page 1-36)
eventlog.enable.list() (on page 1-38)

eventlog.disable.severity

This attribute configures the lowest error or event severity that will be logged.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes			

Usage

```
severity = eventlog.disable.severity
eventlog.disable.severity = severity
```

severity	The highest severity error or event number that will be suppressed.

Details

Set this attribute to any nonzero value to suppress logging of all errors and events with a severity at or below the given severity.

Example

```
print(eventlog.disable.severity)
```

Displays the lowest event severity that will be logged.

Also see

None

eventlog.disable.type

This attribute configures error and event types that will not be logged.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes			

Usage

```
type = eventlog.disable.type
eventlog.disable.type = type
```

type	The types of errors or events that will not be logged

Details

Setting attribute to any nonzero value will suppress logging of all errors and events with a type that matches one of the types in this mask.

Example

`print(eventlog.disable.type)`
Displays event type mask for error events that will not be logged.

Also see

None

eventlog.enable.add()

This function will add a list of errors to the enabled list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.enable.add(list)
```

list	This can be a string in list notation or a table

Details

This command adds the error/event numbers in the list to the `eventlog.enable.list` (on page 1-38) list.

Example

Also see

eventlog.enable.list (on page 1-38)

eventlog.enable.delete()

This function will remove a list of errors from the enabled list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
results = group.function(value)
```

list	This can be a string in list notation or a table

Details

This command removes the error and event numbers in the list from the `eventlog.enable.list` (on page 1-38) list.

Example

Also see

eventlog.enable.list (on page 1-38)

eventlog.enable.list

This attribute stores the list of enabled error and event numbers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes			

Usage

```
list = eventlog.enable.list
eventlog.enable.list = list
```

`list`	You can set this attribute as a string in list notation or a table. When the attribute is read, it will be a string in list notation

Details

This attribute stores the list of disabled error and event numbers. Errors and events in this list will not be logged when encountered. Setting this attribute also affects the `eventlog.disable.list` (on page 1-34) attribute.

Example

```
print(eventlog.enable.list)
```

Displays the list of enabled error and event numbers.

Also see

eventlog.enable.add() (on page 1-36)
eventlog.enable.delete() (on page 1-37)
eventlog.disable.list (on page 1-34)

eventlog.get()

This function performs a non-destructive read. It retrieves one error without removing it from the queue.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

Details

TBD

Example

Also see

None

eventlog.getcount()

This function returns the number of unread events in the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.getcount()
eventlog.getcount(eventType)
```

eventType	Limits the return to specific event log types

Details

This function can be used to retrieve the number of errors or events in the event log of a specific type. `Eventlog.count` (on page 1-32) can also be used to get the total count of all events in the event log.

Example

`print(eventlog.getcount())`
Display the total count of all events in the event log.

Also see

eventlog.count (on page 1-32)

eventlog.gethighseverity()

This function returns the highest severity value of any event in the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
severity = eventlog.gethighseverity()
```

severity	The parameter meaning something***

Details

TBD

Example

```
print(eventlog.gethighseverity())
```

Returns the highest severity value for any event in the event log.

Also see

None

eventlog.next()

This function reads and returns the oldest unread event message from the event log, and is then removed from the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventNumber, message, severity, nodeID, timeSeconds, timeFractional =
    eventlog.next()
eventNumber, message, severity, nodeID, timeSeconds, timeFractional =
    eventlog.next(eventType)
```

eventNumber	The event number
message	A description of the event
severity	The severity of the event
nodeID	The TSP-Link node in which the event occurred
timeSeconds	The seconds portion of the time the event occurred
timeFractional	The fractional seconds portion of the time when the event occurred
eventType	Limits the return to specific event log types

Details

When an event occurs on the instrument, it is placed in the event log in a first-in, first-out (FIFO) queue. This command retrieves the oldest unread event from the event log and removes it from the event log.

If there are no entries in the event log, the following is returned:

```
0 No error 0 0 0 0
```

To read multiple events, execute this command multiple times.

If the event type is not defined, an event of any type is returned.

***********TODO: Consider explaining what happens to events if the queue overflows.

Example

```
print(eventlog.disable.type)
```

Displays event type mask for error events that will not be logged.

Also see

eventlog.post()

This function allows users to post their own text to the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.post(message)
eventlog.post(message, severity)
```

message	String that contains the message
severity	The severity of the event

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. If *`severity`* is not given, `10` will be used.

For compatibility with TTI, this command will accept an error type for severity. If the type is `eventlog.TYPE_INFO`, `10` will be used. If the type is `eventlog.TYPE_WARN`, `15` will be used. If the type is `eventlog.TYPE_ERROR`, `20` will be used. Integer values less then `5` will be treated as an error type. The values `0` and `3` are not allowed. Values greater than `4` are treated as a severity.

Example

Also see

None

eventlog.save()

This function saves the event log to a file on a USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventlog.save(filename)
eventlog.save(filename, eventType)
```

filename	A string that represents the name of the file to be saved.
eventType	The type of event, if no event is defined. Default is TBD.

Details

TBD

Example

Also see

None

eventlog.storemask()

This function filters events from entering the event log. ********This will be supplanted by the eventlog.disable.type attribute

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
********results = group.function(value)
```

****value	

Details

******ki.eventlog.storemask(<type mask>, <severity mask>) is the TTI backdoor function. See TREB-813 for information on Severity VS Type.

Example

Also see

None

eventlog.suppress()

This function suppresses a popup message from the front screen.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
results = group.function(value)
```

value	The parameter meaning something
results	The return code from the function call

Details

TBD

Example

Also see

None

eventlog.unsuppress()

This function unsuppresses a popup from the front screen.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
results = group.function(value)
```

value	The parameter meaning something
results	The return code from the function call

Details

TBD

***Add details here

Example

Also see

None

eventlog.getsuppressed()

This function returns all events that are suppressed from appearing on the front screen.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
results = group.function(value)
```

value	The parameter meaning something
results	The return code from the function call

Details

TBD

Example

Also see

None

fileVar:close()

This function closes the file that is represented by the *fileVar* variable.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
fileVar:close()
```

fileVar	The file descriptor variable to close

Details

This command is equivalent to `io.close(fileVar)`.

Note that files are automatically closed when the file descriptors are garbage collected.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

print("\n*** fileVar:close")
do
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1")
myfile:close()
end
myfile, myfile_err, myfile_errnum = io.open(fileName, "r")
myfile:close()
os.remove(fileName)
```

Opens file `myfile.txt` for writing. If no errors were found while opening, writes `Removing file` and closes the file.

Also see

fileVar:flush() (on page 1-47)
fileVar:read() (on page 1-49)
fileVar:seek() (on page 1-51)
fileVar:write() (on page 1-53)
io.close() (on page 1-67)
io.open() (on page 1-69)

fileVar:flush()

This function writes buffered data to a file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

`fileVar``:flush()`

`fileVar`	The file descriptor variable to flush

Details

The *fileVar*`:write()` or `io.write()` functions buffer data, which may not be written immediately to the USB flash drive. Use *fileVar*`:flush()` to flush this data. Using this function removes the need to close a file after writing to it, allowing the file to be left open to write more data. Data may be lost if the file is not closed or flushed before a script ends.

If there is going to be a time delay before more data is written to a file, and you want to keep the file open, flush the file after you write to it to prevent loss of data.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()
print("\n*** io.read")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1\n")
myfile:flush()
myfile:close()
do
fileHandle = io.input(fileName)
value = io.read("*a")
print(value)
end
fileHandle:close()

print(errorqueue.next())
```

Writes data to a USB flash drive.

Also see

fileVar:write() (on page 1-53)
io.open() (on page 1-69)
io.write() (on page 1-74)

fileVar:read()

This function reads data from a file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
data1 = fileVar:read()
data1 = fileVar:read(format1)
data1, data2 = fileVar:read("format1", "format2")
data1, ..., datan = fileVar:read("format1", ..., "formatn")
```

data1	First data read from the file
data2	Second data read from the file
datan	Last data read from the file
fileVar	The descriptor of the file to be read
format1	A string or number indicating the first type of data to be read
format2	A string or number indicating the second type of data to be read
formatn	A string or number indicating the last type of data to be read
`...`	One or more entries (or values) separated by commas

Details

The format parameters may be any of the following:

`"*n"`: Returns a number.

`"*a"`: Returns the whole file, starting at the current position (returns an empty string if the current file position is at the end of the file).

`"*l"`: Returns the next line, skipping the end of line; returns `nil` if the current file position is at the end of file.

n: Returns a string with up to *n* characters; returns an empty string if n is zero; returns `nil` if the current file position is at the end of file.

If no format parameters are provided, the function performs as if the function is passed the value `"*l"`.

Any number of format parameters may be passed to this command, each corresponding to a returned data value.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

print("fileVar:read")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1")
myfile:close()
do
myfile, myfile_err, myfile_errnum = io.open(fileName, "r")
contents = myfile:read("*a")
print(contents)
end
myfile:close()
os.remove(fileName)
```

Reads data from the input file.

Also see

fileVar:write() (on page 1-53)
io.input() (on page 1-68)
io.open() (on page 1-69)

fileVar:seek()

This function sets and gets the present position of a file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
position, errorMsg = fileVar:seek()
position, errorMsg = fileVar:seek("whence")
position, errorMsg = fileVar:seek("whence", offset)
```

position	The new file position, measured in bytes from the beginning of the file
errorMsg	A string containing the error message
fileVar	The file descriptor variable
whence	A string indicating the base against which *offset* is applied; the default is `"cur"`
offset	The intended new position, measured in bytes from a base indicated by *whence* (default is `0`)

Details

The *whence* parameters may be any of the following:

`"set"`: Beginning of file

`"cur"`: Current position

`"end"`: End of file

If an error is encountered, it is logged to the error queue, and the command returns `nil` and the error string.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()

print("\n*** fileVar:seek")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1")
myfile:close()
do
myfile, myfile_err, myfile_errnum = io.open(fileName, "r")
position = myfile:seek("end", -1)
print(position)
end
myfile:close()
os.remove(fileName)
```

Get the present position of a file.

Also see

io.open() (on page 1-69)

fileVar:write()

This function writes data to a file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
fileVar:write(data)
fileVar:write(data1, data2)
fileVar:write(data1, ..., datan)
```

fileVar	The file descriptor variable
data	Write all data to the file
data1	The first data to write to the file
data2	The second data to write to the file
datan	The last data to write to the file
...	One or more entries (or values) separated by commas

Details

This function may buffer data until a flush (*fileVar*`:flush()` or `io.flush()`) or close (*fileVar*`:close()` or `io.close()`) operation is performed.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()

print("\n*** fileVar:write")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
do
myfile:write("Line 1")
end
myfile:close()
os.remove(fileName)
```

Write data to a file.

Also see

fileVar:close() (on page 1-46)
fileVar:flush() (on page 1-47)
io.close() (on page 1-67)
io.flush() (on page 1-67)
io.open() (on page 1-69)

firmware.update()

This function updates the instrument with the firmware image downloaded with ******flash/endflash

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
firmware.update()
```

Details

This command updates the firmware and reboots the instrument.

*****check wording*********Before issuing this command, the firmware image must be downloaded to the instrument using the ***flash and ***endflash messages. To download the image, first send the flash command. Then send every line of the firmware image .x file each as a separate message. Finally, send the endflash command.

Example

Also see

None

format.asciiprecision

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Instrument reset Recall setup	Not saved	6

Usage

```
precision = format.asciiprecision
format.asciiprecision = precision
```

precision	A number representing the number of digits to be printed for numbers printed with the `print()`, `printbuffer()`, and `printnumber()` functions; must be a number between 0 and 16. The default is 6.

Details

This attribute specifies the precision (number of digits) for numeric data printed with the `print()`, `printbuffer()`, and `printnumber()` functions. The attribute is only used with the ASCII format. The precision value must be a number from 0 to 16. When the precision is set to 0, the instrument uses the native Lua formatting of numbers.

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

Example

```
format.asciiprecision = 0
x = 2.54
printnumber(x)
format.asciiprecision = 0
printnumber(x)
```

Output:

```
2.54000000000e+00

2.54e+00
```

Also see

format.byteorder (on page 1-55)
format.data (on page 1-57)

format.byteorder

This attribute sets the binary byte order for the data that is printed using the `printnumber()` and `printbuffer()` functions.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Instrument reset Recall setup	Not saved	1 (format.LITTLEENDIAN)

Usage

```
order = format.byteorder
format.byteorder = order
```

order	Byte order value as follows: ■ Most significant byte first: `0`, `format.NORMAL`, `format.NETWORK`, or `format.BIGENDIAN` ■ Least significant byte first: `1`, `format.SWAPPED` or `format.LITTLEENDIAN`

Details

This attribute selects the byte order in which data is written when you are printing data values with the `printnumber()` and `printbuffer()` functions. The byte order attribute is only used with the `format.SREAL`, `format.REAL`, `format.REAL32`, and `format.REAL64` data formats.

`format.NORMAL`, `format.BIGENDIAN`, and `format.NETWORK` select the same byte order. `format.SWAPPED` and `format.LITTLEENDIAN` select the same byte order. Selecting which to use is a matter of preference.

Select the `format.SWAPPED` or `format.LITTLEENDIAN` byte order when sending data to a computer with a Microsoft Windows operating system.

Example

`x = 1.23`
`format.data = format.REAL32`
`format.byteorder = format.LITTLEENDIAN`
`printnumber(x)`
`format.byteorder = format.BIGENDIAN`
`printnumber(x)`
	The output depends on the terminal program you use, but it looks something like:
	`#0¤p??`
	`#0??p¤`

Also see

format.asciiprecision
format.data (on page 1-57)
printbuffer() (on page 1-101)
printnumber() (on page 1-102)

format.data

This attribute sets the data format for data that is printed using the printnumber() and printbuffer() functions.

Usage

```
value = format.data
format.data = value
```

value	The format to use for data, set to one of the following values: ■ ASCII format: `1` or `format.ASCII` ■ Single-precision IEEE Std 754 binary format: 2, `format.SREAL`, or `format.REAL32` ■ Double-precision IEEE Std 754 binary format: 3, `format.REAL`, `format.REAL64`, or `format.DREAL`

Details

The precision of numeric values can be controlled with the `format.asciiprecision` attribute. The byte order of `format.SREAL`, `format.REAL`, `format.REAL32`, and `format.REAL64` can be selected with the `format.byteorder` attribute.

REAL32 and SREAL select the same single precision format. REAL and REAL64 select the same double-precision format. They are alternative identifiers. Selecting which to use is a matter of preference.

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.

NOTE

Binary formats are not intended to be interpreted by humans.

Example

`format.asciiprecision = 10`
`x = 3.14159265`
`format.data = format.ASCII`
`printnumber(x)`
`format.data = format.REAL64`
`printnumber(x)`
	Output a number represented by *x* in ASCII using a precision of 10, then output the same number in binary using double-precision format.
	Output:
	`3.141592650e+00`
	`#0ñÔÈSû!   @`

Also see

format.asciiprecision
format.byteorder (on page 1-55)
printbuffer() (on page 1-101)
printnumber() (on page 1-102)

fs.chdir()

This function sets the current working directory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`workingDirectory` `= fs.chdir("`*`path`*`")`

`workingDirectory`	Returned value containing the working path
`path`	A string indicating the new working directory path

Details

The new working directory path may be absolute or relative to the current working directory.

An error is logged to the error queue if the given path does not exist.

Example

```
if fs.is_dir("/usb1/temp") == true then
  fs.chdir("/usb1/temp")
  testPath = fs.cwd()
  print(testPath)
else
  testPath = fs.cwd()
  print(testPath)
end
```

Insert a USB flash drive into the front panel of the instrument.
Verify that `/usb1/temp` is a directory and change it to be the current working directory.
Set the variable for the current working directory to be `testPath`.
The return should be:
`/usb1/temp`
If `/usb1/temp` is not a directory, set the variable for the current working directory to be `testPath`.
The return is:
`/usb1`

Also see

None

fs.cwd()

This function returns the absolute path of the current working directory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

path = `fs.cwd()`

path	The absolute path of the current working directory

Example

```
if fs.is_dir("/usb1/temp") == true then
  fs.chdir("/usb1/temp")
  testPath = fs.cwd()
  print(testPath)
else
  testPath = fs.cwd()
  print(testPath)
end
```

Insert a USB flash drive into the front panel of the instrument.
Verify that `/usb1/temp` is a directory and change it to be the current working directory.
Set the variable for the current working directory to be `testPath`.
The return should be:

```
/usb1/temp
```

If `/usb1/temp` is not a directory, set the variable for the current working directory to be `testPath`.
The return is:

```
/usb1
```

Also see

None

fs.is_dir()

This function tests whether or not the specified path refers to a directory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status = fs.is_dir("path")
```

status	Whether or not the given path is a directory (`true` or `false`)
path	The path of the file system entry to test

Details

The file system path may be absolute or relative to the current working system path.

Example 1

```
print("Is directory: ", fs.is_dir("/usb1/"))
```

Because `/usb1/` is always the root directory of an inserted flash drive, you can use this command to verify that USB flash drive is inserted.

Example 2

```
if fs.is_dir("/usb1/temp") == false then
    fs.mkdir("/usb1/temp")
end
```

Insert a USB flash drive into the front panel of the instrument.
Check to see if the `temp` directory exists.
If it does not exist, create a directory named `temp`.

Also see

fs.is_file() (on page 1-61)

fs.is_file()

Tests whether the specified path refers to a file (as opposed to a directory).

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status = fs.is_file("path")
```

`status`	`true` if the given path is a file; otherwise, `false`
`path`	The path of the file system entry to test

Details

The file system path may be absolute or relative to the current working system path.

Example

```
rootDirectory = "/usb1/"
print("Is file: ", fs.is_file(rootDirectory))
```

Insert a USB flash drive into the front panel of the instrument.
Set `rootDirectory` to be the USB port.
Check to see if `rootDirectory` is a file. Because `rootDirectory` was set up as a directory, the return is `false`.

Also see

fs.is_dir() (on page 1-60)

fs.mkdir()

This function creates a directory at the specified path.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

path = `fs.mkdir("`*newPath*`")`

path	The returned path of the new directory
newpath	Location (path) of where to create the new directory

Details

The directory path may be absolute or relative to the current working directory.

An error is logged to the error queue if the parent folder of the new directory does not exist, or if a file system entry already exists at the given path.

Example

```
if fs.is_dir("/usb1/temp") == false then
    fs.mkdir("/usb1/temp")
end
```

Insert a USB flash drive into the front panel of the instrument.
Check to see if the `temp` directory exists.
If it does not exist, create a directory named `temp`.

Also see

fs.rmdir() (on page 1-64)

fs.readdir()

This function returns a list of the file system entries in the directory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
files = fs.readdir("path")
```

files	A table containing the names of all the file system entries in the specified directory
path	The directory path

Details

The directory path may be absolute or relative to the current working directory.

This command is nonrecursive. For example, entries in subfolders are not returned.

An error is logged to the error queue if the given path does not exist or does not represent a directory.

Example

```
rootDirectory = "/usb1/"
entries = fs.readdir(rootDirectory)
count = table.getn(entries)
print("Found a total of "..count.." files and directories")
for i = 1, count do
   print(entries[i])
end
```

Insert a USB flash drive into the front panel of the instrument.
Set `rootDirectory` to be the USB port.
Set `entries` as the variable for the file system entries in `rootDirectory`.
Return the number of files and directories in the directory.

Also see

None

fs.rmdir()

This function removes a directory from the file system.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
fs.rmdir("path")
```

path	The path of the directory to remove

Details

This path may be absolute or relative to the current working directory.

An error is logged to the error queue if the given path does not exist or does not represent a directory. An error is also logged if the directory is not empty.

Example

```
rootDirectory = "/usb1/"
tempDirectoryName = "temp"
if fs.is_dir(rootDirectory..tempDirectoryName) == false then
    fs.mkdir(rootDirectory..tempDirectoryName)
end
fs.rmdir(rootDirectory..tempDirectoryName)
```

Insert a USB flash drive into the front panel of the instrument.
Set `rootDirectory` to be the USB port.
Set `tempDirectoryName` to be equivalent to `temp`.
Check to see if `tempDirectoryName` exists.
If it does not exist, create a directory named `temp`.
Remove the directory.

Also see

fs.mkdir() (on page 1-62)

gettimezone()

This function retrieves the local time zone.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
timeZone = gettimezone()
```

timeZone	The local time zone of the instrument

Details

See `settimezone()` for additional details about the time zone format and a description of the fields.

`timeZone` can be in either of the following formats:

- If one parameter was used with `settimezone()`, the format used is:
  `GMThh:mm:ss`
- If four parameters were used with `settimezone()`, the format used is:
  `GMThh:mm:ssGMThh:mm:ss,Mmm.w.dw/hh:mm:ss,Mmm.w.dw/hh:mm:ss`

Example

`timezone = gettimezone()`
Reads the value of the local time zone.

Also see

settimezone() (on page 1-104)

info

Retrieves system information about the mainframe and connected slots.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
systeminfo = info()
```

Details

This command returns system information as a string for the mainframe and connected modules.

On the mainframe side, information returned will be all manufactured strings saved to nonvolatile memory using the `ki.sysinfo.addpair()` command, lan settings, OS version, BSP version, and firmware revision.

On the module side, information returned will be all manufactured strings saved to nonvolatile memory using the `ki.sysinfo.addpair()` command, FPGA versions, number of module interfaces connected in the slot, whether it requires 48 V, and firmware revision. This information is for each interface (upper and lower) for the slot.

Example

Also see

io.close()

This function closes a file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes (see **Details**)			

Usage

```
io.close()
io.close(file)
```

file	The descriptor of the file to close

Details

If a file is not specified, the default output file closes.

Only `io.close()`, used without specifying a parameter, can be accessed from a remote node.

Example

```
testFile, testError = io.open("testfile.txt", "w")
if nil == testError then
   testFile:write("This is my test file")
   io.close(testFile)
end
```

Opens file `testfile.txt` for writing. If no errors were found while opening, writes "`This is my test file`" and closes the file.

Also see

io.open() (on page 1-69)

io.flush()

This function saves buffered data to a file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
io.flush()
```

Details

You must use the `io.flush()` or `io.close()` functions to write data to the file system.

NOTE

Data is not automatically written to a file when you use the `io.write()` function. The `io.write()` function buffers data; it may not be written to the USB flash drive immediately. Use the `io.flush()` function to immediately write buffered data to the drive.

This function only flushes the default output file.

Using this command removes the need to close a file after writing to it and allows it to be left open to write more data. Data may be lost if the file is not closed or flushed before an application ends. To prevent the loss of data if there is going to be a time delay before more data is written (and when you want to keep the file open and not close it), flush the file after writing to it.

Also see

fileVar:flush() (on page 1-47)
fileVar:write() (on page 1-53)
io.write() (on page 1-74)

io.input()

This function assigns a previously opened file, or opens a new file, as the default input file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes (see **Details**)			

Usage

```
fileVar = io.input()
fileVar = io.input("newfile")
```

fileVar	The descriptor of the input file or an error message (if the function fails)
newfile	A string representing the path of a file to open as the default input file, or the file descriptor of an open file to use as the default input file

Details

The *newfile* path may be absolute or relative to the current working directory.

When using this function from a remote TSP-Link^{TM} node, this command does not accept a file descriptor and does not return a value.

If the function fails, an error message is returned.

Also see

io.open() (on page 1-69)
io.output() (on page 1-70)

io.open()

This function opens a file for later reference.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
fileVar, errorMsg = io.open("path")
fileVar, errorMsg = io.open("path", "mode")
```

fileVar	The descriptor of the opened file
errorMsg	Indicates whether an error was encountered while processing the function
path	The path of the file to open
mode	A string representing the intended access mode (`"r"` = read, `"w"` = write, and `"a"` = append)

Details

The path to the file to open may be absolute or relative to the current working directory. If you successfully open the file, *errorMsg* is nil and *fileVar* has the descriptor used to access the file.

If an error is encountered, the command returns `nil` for *fileVar* and an error string.

Example

```
testFile, testError = io.open("testfile.txt", "w")
if testError == nil then
   testFile:write("This is my test file")
   io.close(testFile)
end
```

Opens file `testfile.txt` for writing. If no errors were found while opening, writes `"This is my test file"` and closes the file.

Also see

io.close() (on page 1-67)

io.output()

This function assigns a previously opened file or opens a new file as the default output file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes (see **Details**)			

Usage

```
fileVar = io.output()
fileVar = io.output("newfile")
```

fileVar	The descriptor of the output file or an error message (if the function fails)
newfile	A file descriptor to assign (or the path of a file to open) as the default output file

Details

The path of the file to open may be absolute or relative to the current working directory.

When accessed from a remote node using the TSP-Link network, this command does not accept a file descriptor parameter and does not return a value.

If the function fails, an error message is returned.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()

print("\n*** io.output")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1")
myfile:close()
do
fileHandle = io.output(fileName)
print(fileHandle)
end
io.close(fileHandle)
print(fileHandle)
os.remove(fileName)
```

Assign the file to be the default output file.

Also see

io.input() (on page 1-68)
io.open() (on page 1-69)

io.read()

This function reads data from the default input file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
data1 = io.read()
data1 = io.read("format1")
data1, data2 = io.read("format1", "format2")
data1, ..., dataN = io.read("format1", ..., "formatN")
```

data1	The data read from the file
data2	The data read from the file
dataN	The data read from the file; the number of return values matches the number of format values given
format1	A string or number indicating the type of data to be read
format2	A string or number indicating the type of data to be read
formatN	A string or number indicating the type of data to be read
`...`	One or more entries (or values) separated by commas

Details

The format parameters may be any of the following:

Format parameter	Description
`"*n"`	Returns a number
`"*a"`	Returns the whole file, starting at the present position; returns an empty string if it is at the end of file
`"*l"`	Default setting; returns the next line, skipping the end of line; returns `nil` if the present file position is at the end of file
N	Returns a string with up to *N* characters; returns an empty string if *N* is zero (0); returns `nil` if the present file position is at the end of file

Any number of format parameters may be passed to this command, each corresponding to a returned data value.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()

-- io.read
print("\n*** io.read")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1\n")
myfile:flush()
myfile:close()
do
fileHandle = io.input(fileName)
value = io.read("*a")
print(value)
end
fileHandle:close()

print(errorqueue.next())
```

Read data from the default input file.

Also see

None

io.type()

This function checks whether or not a given object is a file handle.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
type = io.type(obj)
```

type	Indicates whether the object is an open file handle
obj	Object to check

Details

Returns the string `"file"` if the object is an open file handle. If it is not an open file handle, `nil` is returned.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()

print("\n*** io.type")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1")
myfile:close()
do
fileHandle = io.output(fileName)
state = io.type(fileHandle)
print(state)
end
io.close(fileHandle)
local state = io.type(fileHandle)
print(state)
os.remove(fileName)
```

Check whether or not `fileName` is a file handle.

Also see

io.open() (on page 1-69)

io.write()

This function writes data to the default output file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
io.write()
io.write(data1)
io.write(data1, data2)
io.write(data1, ..., dataN)
```

data1	The data to be written
data2	The data to be written
dataN	The data to be written
...	One or more values separated by commas

Details

All data parameters must be either strings or numbers.

NOTE

Data is not immediately written to a file when you use the `io.write()` function. The `io.write()` function buffers data; it may not be written to the USB flash drive immediately. Use the `io.flush()` function to immediately write buffered data to the drive.

Example

```
local fileName = "/usb1/myfile.txt"

if fs.is_file(fileName) then
   os.remove(fileName)
   print("Removing file")
else
   print("Nothing removed")
end

errorqueue.clear()

print("\n*** io.write")
myfile, myfile_err, myfile_errnum = io.open(fileName, "w")
myfile:write("Line 1")
myfile:close()
do
fileHandle = io.output(fileName)
io.write("Line 2")
end
io.close(fileHandle)
os.remove(fileName)
```

Writes data to the default output file.

Also see

io.flush() (on page 1-67)

lan.applysettings()

This function re-initializes the LAN interface with new settings.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
lan.applysettings()
```

Details

Disconnects all existing LAN connections to the instrument and re-initializes the LAN with the present configuration settings.

This function initiates a background operation. LAN configuration could be a lengthy operation. Although the function returns immediately, the LAN initialization continues to run in the background.

Even though the LAN configuration settings may not have changed since the LAN was last connected, new settings may take effect due to the dynamic nature of dynamic host configuration protocol (DHCP) or dynamic link local addressing (DLLA) configuration.

Re-initialization takes effect even if the configuration has not changed since the last time the instrument connected to the LAN.

Example

```
lan.applysettings()
```

Re-initialize the LAN interface with new settings.

Also see

None

lan.config.dns.address[N]

Configures DNS server IP addresses.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	"0.0.0.0"

Usage

```
dnsAddress = lan.config.dns.address[N]
lan.config.dns.address[N] = "dnsAddress"
```

dnsAddress	DNS server IP address
N	Entry index (1 or 2)

Details

This attribute is an array of Domain Name System (DNS) server addresses. These addresses take priority for DNS lookups and are consulted before any server addresses that are obtained using DHCP. This allows local DNS servers to be specified that take priority over DHCP-configured global DNS servers.

You can specify up to two addresses. The address specified by `1` is consulted first for DNS lookups. *`dnsAddress`* must be a string specifying the IP address of the DNS server in dotted decimal notation.

Unused entries are returned as `"0.0.0.0"` when read. To disable an entry, set its value to `"0.0.0.0"` or the empty string `""`.

Although only two addresses may be manually specified here, the instrument uses up to three DNS server addresses. If two are specified here, only one that is given by a DHCP server is used. If no entries are specified here, up to three addresses that are given by a DHCP server are used.

Example

```
dnsaddress = "164.109.48.173"
lan.config.dns.address[1] = dnsaddress
```

Set the DNS address 1 to `164.109.48.173`.

Also see

lan.config.dns.domain (on page 1-78)
lan.config.dns.dynamic (on page 1-79)
lan.config.dns.hostname (on page 1-80)
lan.config.dns.verify (on page 1-81)
lan.restoredefaults() (on page 1-88)

lan.config.dns.domain

Configures the dynamic DNS domain.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	""

Usage

```
domain = lan.config.dns.domain
lan.config.dns.domain = "domain"
```

domain	Dynamic DNS registration domain; use a string of 255 characters or less

Details

This attribute holds the domain to request during dynamic DNS registration. Dynamic DNS registration works with DHCP to register the domain specified in this attribute with the DNS server.

The length of the fully qualified host name (combined length of the domain and host name with separator characters) must be less than or equal to 255 characters. Although up to 255 characters are allowed, you must make sure the combined length is also no more than 255 characters.

Example

```
print(lan.config.dns.domain)
```

Outputs the present dynamic DNS domain. For example, if the domain is "Matrix", the response is:

```
Matrix
```

Also see

lan.config.dns.dynamic (on page 1-79)
lan.config.dns.hostname (on page 1-80)
lan.config.dns.verify (on page 1-81)
lan.restoredefaults() (on page 1-88)

lan.config.dns.dynamic

Enables or disables the dynamic DNS registration.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	1 (lan.ENABLE)

Usage

```
state = lan.config.dns.dynamic
lan.config.dns.dynamic = state
```

state	The dynamic DNS registration state. It may be one of the following values: ▪ `1` or `lan.ENABLE`: Enabled ▪ `0` or `lan.DISABLE`: Disabled

Details

Dynamic DNS registration works with DHCP to register the host name with the DNS server. The host name is specified in the `lan.config.dns.hostname` attribute.

Example

```
print(lan.config.dns.dynamic)
```

Outputs the dynamic registration state.
If dynamic DNS registration is enabled, the response is:
```
1.0000000e+00
```

Also see

lan.config.dns.hostname (on page 1-80)
lan.restoredefaults() (on page 1-88)

lan.config.dns.hostname

This attribute defines the dynamic DNS host name.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Not applicable	Nonvolatile memory	Instrument specific (see **Details**)

Usage

```
hostName = lan.config.dns.hostname
lan.config.dns.hostname = "hostName"
```

hostName	The host name to use for dynamic DNS registration; the host name must be a string of ■ ■ 63 characters or less ■ start with a letter ■ end with a letter or digit ■ contain only letters, digits, and hyphens

Details

This attribute holds the host name to request during dynamic DNS registration. Dynamic DNS registration works with DHCP to register the host name specified in this attribute with the DNS server.

The factory default value for *`hostName`* is "

`k-<model number>-<serial number>`", where `<model number>` and `<serial number>` are replaced with the actual model number and serial number of the instrument (for example, "`k-MP5000-1234567`"). Note that hyphens separate the characters of *`hostName`*.

The length of the fully qualified host name (combined length of the domain and host name with separator characters) must be less than or equal to 255 characters. Although up to

63 characters can be entered here, you must make sure the combined length is no more than 255 characters.

Setting this attribute to an empty string (in other words, setting this attribute to a string of length zero or a string that consists entirely of whitespace characters) reverts the host name to the factory default value.

Example

`print(lan.config.dns.hostname)`
Outputs the present dynamic DNS host name.

Also see

None

lan.config.dns.verify

This attribute defines the DNS host name verification state.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	1 (lan.ENABLE)

Usage

```
state = lan.config.dns.verify
lan.config.dns.verify = state
```

state	DNS hostname verification state: ■ `1` or `lan.ENABLE`: DNS host name verification enabled ■ `0` or `lan.DISABLE`: DNS host name verification disabled

Details

When this is enabled, the instrument performs DNS lookups to verify that the DNS host name matches the value specified by `lan.config.dns.hostname`.

Example

```
print(lan.config.dns.verify)
```

Outputs the present DNS host name verification state.
If it is enabled, the output is:

```
1.0000000e+00
```

Also see

lan.config.dns.hostname (on page 1-80)
lan.restoredefaults() (on page 1-88)

lan.config.gateway

This attribute contains the LAN default gateway address.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	"0.0.0.0"

Usage

```
gatewayAddress = lan.config.gateway
lan.config.gateway = "gatewayAddress"
```

gatewayAddress	LAN default gateway address; must be a string specifying the default IP address of the gateway in dotted decimal notation

Details

This attribute specifies the default gateway IP address to use when manual or DLLA configuration methods are used to configure the LAN. If DHCP is enabled, this setting is ignored.

This attribute does not indicate the actual setting that is presently in effect. Use the `lan.status.gateway` attribute to determine the present operating state of the LAN.

The IP address must be formatted in four groups of numbers, each separated by a decimal.

Example

```
print(lan.config.gateway)
```

Outputs the default gateway address. For example, you might see the output:

```
192.168.0.1
```

Also see

lan.restoredefaults() (on page 1-88)
lan.status.gateway (on page 1-90)

lan.config.ipaddress

This command specifies the LAN IP address.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	"192.168.0.2"

Usage

```
ipAddress = lan.config.ipaddress
lan.config.ipaddress = "ipAddress"
```

ipAddress	LAN IP address; must be a string specifying the IP address in dotted decimal notation

Details

This command specifies the LAN IP address to use when the LAN is configured using the manual configuration method. This setting is ignored when DLLA or DHCP is used.

This attribute does not indicate the actual setting that is presently in effect. Use the `lan.status.ipaddress` attribute to determine the present operating state of the LAN.

Example

```
ipaddress = lan.config.ipaddress
```

Retrieves the presently set LAN IP address.

Also see

lan.restoredefaults() (on page 1-88)

lan.restoredefaults() (on page 1-88)
lan.status.ipaddress (on page 1-90)

lan.config.method

This attribute contains the LAN settings configuration method.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	0 (lan.AUTO)

Usage

```
method = lan.config.method
lan.config.method = method
```

method	The method for configuring LAN settings; it can be one of the following values: ■ `0` or `lan.AUTO`: Selects automatic sequencing of configuration methods ■ `1` or `lan.MANUAL`: Use only manually specified configuration settings

Details

This attribute controls how the LAN IP address, subnet mask, default gateway address, and DNS server addresses are determined.

When method is `lan.AUTO`, 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, it uses the manually specified settings.

When method is `lan.MANUAL`, only the manually specified settings are used. Neither DHCP nor DLLA are attempted.

Example

```
print(lan.config.method)
```

Outputs the present method.
For example:
```
1.0000000e+00
```

Also see

lan.restoredefaults() (on page 1-88)

lan.config.subnetmask

This attribute contains the LAN subnet mask.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	"255.255.255.0"

Usage

```
mask = lan.config.subnetmask
lan.config.subnetmask = "mask"
```

mask	String that specifies the LAN subnet mask value in dotted decimal notation

Details

This attribute specifies the LAN subnet mask that is used when the manual configuration method is used to configure the LAN. This setting is ignored when DLLA or DHCP is used.

This attribute does not indicate the actual setting presently in effect. Use the `lan.status.subnetmask` attribute to determine the present operating state of the LAN.

Example

```
print(lan.config.subnetmask)
```

Outputs the LAN subnet mask, such as:

```
255.255.255.0
```

Also see

lan.restoredefaults() (on page 1-88)
lan.status.subnetmask (on page 1-91)

lan.ipconfig()

This function specifies the LAN configuration for the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No	LAN reset	Nonvolatile memory	lan.MODE_AUTO

Usage

```
method, ipV4Address, subnetMask, gateway = lan.ipconfig()
lan.ipconfig(method)
lan.ipconfig(method, "ipV4Address")
lan.ipconfig(method, "ipV4Address", "subnetMask")
lan.ipconfig(method, "ipV4Address", "subnetMask", "gateway")
```

method	The method for configuring LAN settings; it can be one of the following values: ■ The instrument automatically assigns LAN settings: `lan.MODE_AUTO` ■ Manually specify the LAN settings: `lan.MODE_MANUAL`
ipV4Address	LAN IP address; must be a string specifying the IP address in dotted decimal notation
subnetMask	The LAN subnet mask; must be a string in dotted decimal notation
gateway	The LAN default gateway; must be a string in dotted decimal notation

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.

Example

```
lan.ipconfig(lan.MODE_AUTO)
print(lan.ipconfig())
lan.ipconfig(lan.MODE_MANUAL, "192.168.0.7", "255.255.240.0", "192.168.0.3")
print(lan.ipconfig())
```

Set the IP configuration method to automatic. Request the IP configuration. Example output:

```
lan.MODE_AUTO 134.63.78.136   255.255.254.0   134.63.78.1
```

Set the IP configuration method to manual. Request the IP configuration. Output:

```
lan.MODE_MANUAL 192.168.0.7    255.255.240.0  192.168.0.3
```

Also see

None

lan.macaddress

This attribute describes the LAN MAC address.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	No	Not applicable	Not applicable	Not applicable

Usage

```
MACaddress = lan.macaddress
```

MACaddress	The MAC address of the instrument

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

`print(lan.macaddress)`	Returns the MAC address. For example: `08:00:11:00:00:57`

Also see

lan.ipconfig() (on page 1-86)

lan.restoredefaults()

This function resets LAN settings to default values.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
lan.restoredefaults()
```

Details

The settings that are restored are shown in the following table.

Settings that are restored to default	
Attribute	**Default setting**
`lan.autoconnect`	`lan.ENABLE`
`lan.config.dns.address[N]`	`"0.0.0.0"`
`lan.config.dns.domain`	`""`
`lan.config.dns.dynamic`	`lan.ENABLE`
`lan.config.dns.hostname`	`"K-<model number>-<serial number>"`
`lan.config.dns.verify`	`lan.ENABLE`
`lan.config.duplex`	`lan.FULL`
`lan.config.gateway`	`"0.0.0.0"`
`lan.config.ipaddress`	`"0.0.0.0"`
`lan.config.method`	`lan.AUTO`
`lan.config.speed`	`100`
`lan.config.subnetmask`	`"255.255.255.0"`
`lan.linktimeout`	`20` (seconds)
`lan.lxidomain`	`0`
`lan.nagle`	`lan.DISABLE`
`lan.timedwait`	`20` (seconds)

This command is run when `lan.reset()` is sent.

Example

`lan.restoredefaults()`
Restores the LAN defaults.

Also see

lan.reset()
localnode.password

lan.status.dns.name

This attribute contains the present DNS fully qualified host name.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
hostName = lan.status.dns.name
```

hostName	Fully qualified DNS host name that can be used to connect to the instrument

Details

A fully qualified domain name (FQDN) specifies its exact location in the tree hierarchy of the Domain Name System (DNS).

A FQDN is the complete domain name for a specific computer or host on the LAN. The FQDN consists of two parts: The host name and the domain name.

If the DNS host name for an instrument is not found, this attribute stores the IP address in dotted decimal notation.

Example

```
print(lan.status.dns.name)
```

Outputs the dynamic DNS host name.

Also see

lan.config.dns.address[N] (on page 1-77)
lan.config.dns.hostname (on page 1-80)

lan.status.gateway

This attribute contains the gateway address presently in use by the LAN interface.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
gatewayAddress = lan.status.gateway
```

gatewayAddress	LAN gateway address presently being used

Details

The value of *gatewayAddress* is a string that indicates the IP address of the gateway in dotted decimal notation.

Example

```
print(lan.status.gateway)
```

Outputs the gateway address, such as:

```
192.168.0.1
```

Also see

lan.config.gateway (on page 1-82)

lan.status.ipaddress

This attribute contains the LAN IP address presently in use by the LAN interface.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
ipAddress = lan.status.ipaddress
```

ipAddress	LAN IP address specified in dotted decimal notation

Details

The IP address is a character string that represents the IP address assigned to the instrument.

Example

```
print(lan.status.ipaddress)
```

Outputs the LAN IP address currently in use, such as:

```
192.168.0.2
```

Also see

lan.config.ipaddress (on page 1-83)

lan.status.method

This attribute describes something.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Never	Nonvolatile memory	1

Usage

```
value = group.attribute
```

value	Describe value

Details

This attribute

Example

Also see

Insert hyperlinks here

lan.status.subnetmask

This attribute contains the LAN subnet mask that is presently in use by the LAN interface.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
mask = lan.status.subnetmask
```

mask	A string specifying the subnet mask in dotted decimal notation

Details

Use this attribute to determine the present operating state of the LAN. This attribute returns the present LAN subnet mask value if the LAN is manually configured, or when DLLA or DHCP is used.

Example

```
print(lan.status.subnetmask)
```

Outputs the subnet mask of the instrument that is presently in use, such as:
`255.255.255.0`

Also see

lan.config.subnetmask (on page 1-85)

localnode.linefreq

This attribute contains the power line frequency detected at power-on.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Nonvolatile memory	60 (60 Hz)

Usage

```
frequency = localnode.linefreq
```

frequency	An integer representing the detected power line frequency of the instrument

Details

This attribute holds the value of the power line frequency.

Some modules achieve optimum noise rejection when performing measurements at integer NPLC apertures. The instrument automatically detects the power line frequency at startup and all modules will use this value when calculating apertures in terms of NPLC.

Example

```
frequency = localnode.linefreq
```

Reads the power line frequency

Also see

None

localnode.manufacturer

This attribute stores the manufacturer of the instrument

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
manufacturer = localnode.manufacturer
```

manufacturer	This will always be `Keithley Instruments`

Details

This attribute stores the manufacturer of the instrument.

Example

```
print(localnode.model)
```

Outputs the manufacturer.

Also see

None

localnode.model

This attribute stores the model number of the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
model = localnode.model
```

model	The model number of the instrument

Details

This attribute stores the model number of the instrument.

Example

```
print(localnode.model)
```

Outputs the model number of the local node.

Also see

localnode.serialno (on page 1-96)

localnode.prompts

This attribute determines if the instrument generates prompts in response to command messages.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Power cycle	Not saved	localnode.DISABLE

Usage

```
prompting = localnode.prompts
localnode.prompts = prompting
```

prompting	Prompting mode ■ `localnode.ENABLE`: Generate prompts ■ `localnode.DISABLE`: Do not generate prompts

Details

When the prompting mode is enabled, the instrument generates prompts when the instrument isready to take another command. Because the prompt is not generated until the previous command completes, enabling prompts provides handshaking with the instrument to prevent buffer overruns.

When prompting is enabled, the instrument might generate the following prompts:

- **TSP>**. The standard prompt, which indicates that the previous command completed normally.
- **TSP?**. The prompt that is issued if there are unread entries in the error queue when the prompt is issued. Like the TSP> prompt, it indicates that processing of the command is complete. It does not mean the previous command generated an error, only that there were still errors in the queue when the command processing was complete.
- **>>>>**. The continuation prompt, which occurs when downloading scripts. When downloading scripts, many command messages must be sent as a group. The continuation prompt indicates that the instrument is expecting more messages as part of the present command.

Commands do not generate prompts. The instrument generates prompts in response to command completion.

Prompts are enabled or disabled only for the remote interface that is active when you send the command. For example, if you enable prompts when the LAN connection is active, they are not enabled for a subsequent USB connection.

Example

`localnode.ENABLE`
Generate prompts.

Also see

localnode.prompts4882 (on page 1-95)
localnote.showerrors (on page 1-97)

localnode.prompts4882

This attribute enables or disables the generation of prompts for IEEE Std 488.2 common commands.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Power cycle	Not saved	localnode.DISABLE

Usage

```
prompting = localnode.prompts4882
localnode.prompts4882 = prompting
```

prompting	Prompting mode ▪ `localnode.ENABLE`: Generate prompts ▪ `localnode.DISABLE`: Do not generate prompts

Details

When this attribute is enabled, the IEEE Std 488.2 common commands generate prompts if prompting is enabled with the `localnode.prompts` attribute. If `localnode.prompts4882` is enabled, limit the number of `*trg` commands sent to a running script to 50 regardless of the setting of the `localnode.prompts` attribute.

When this attribute is disabled, IEEE Std 488.2 common commands do not generate prompts. When using the `*trg` command with a script that executes `trigger.wait()` repeatedly, disable prompting to avoid problems associated with the command interface input queue filling.

Example

`localnode.prompts4882 = ENABLE`
Enables IEEE Std 288.2 common command prompting.

Also see

localnode.prompts (on page 1-94)

localnode.serialno

This attribute stores the serial number of the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
serialno = localnode.serialno
```

serialno	The serial number of the instrument

Details

This attribute stores the serial number of the instrument.

Example

```
display.settext(localnode.serialno)
```

Displays the serial number of the instrument

Also see

localnode.model (on page 1-93)
localnode.version (on page 1-98)

localnode.showerrors

This attribute sets whether or not the instrument automatically sends generated errors.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Power cycle	Not saved	localnode.DISABLE

Usage

```
errorMode = localnode.showerrors
localnode.showerrors = errorModee
```

errorMode	Show error setting: ■ `localnode.ENABLE`: Send errors ■ `localnode.DISABLE`: Do not send errors

Details

If this attribute is set to `1`, the instrument automatically sends any generated errors stored in the error queue, and then clears the queue. Errors are processed after executing a command message (just before issuing a prompt if prompts are enabled).

If this attribute is set to `0`, errors are left in the error queue and must be explicitly read or cleared.

When using this command from a remote node, replace `localnode` with the node reference, for example, `node[5].showerrors`.

Example

`localnode.showerrors = 1`
Enables sending of generated errors.

Also see

localnode.prompts (on page 1-94)
localnote.prompts4882 (on page 1-95)

localnode.version

This attribute stores the firmware revision level.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
version = localnode.version
```

version	The firmware revision number of the instrument

Details

This attribute stores the firmware revision level of the instrument.

Example

```
print(localnode.revision)
```

Outputs the revision level of the instrument

Also see

localnode.manufacturer (on page 1-93)
localnode.model (on page 1-93)
localnode.serialno (on page 1-96)

os.clock()

This function returns the number of seconds the instrument has been powered-on.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
seconds = os.clock()
```

seconds	The number of seconds that have elapsed since the instrument was powered-on

Details

This function returns the total elapsed time since the instrument was powered-on.

Example

```
t = os.clock()
```

Returns the elapsed time since the instrument was powered-on.

Also see

os.time() (on page 1-99)

os.time()

This function generates a time value in UTC time.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
utcSeconds, utcFractional = os.time()
utcSeconds, utcFractional = os.time(timespec)
```

utcSeconds	Time value in UTC time, which is the number of seconds since midnight January 1, 1970 UTC
utcFractional	When using the first form, this value will hold the fractional seconds. When using the second form, this value will always be 0
timespec	The date and time (year, month, day, hour, and minute)

Details

When used without parameters, this function returns current time as the number of seconds since midnight January 1, 1970. The *utcSeconds* value will be the elapsed seconds and *utcFractional* will be the fractional seconds. Both values have the units of seconds. *utcSeconds* will be an integer and *utcFractional* will be a value between `0` and `1` but always less than `1`.

When using the second form, *timespec* is a table using the fields listed below.

year	The year (`1970` or later)
month	The month (`1` to `12`)
day	The day (`1` to `31`)
hour	The hour (`0` to `23`)
min	The minute (`0` to `59`)
sec	The second (`0` to `59`)

If the time (hour, minute, second) fields are not provided, noon on that day will be used. The time specified is given in the local timezone.

Example

```
print(os.date('%Y-%m-%d %H:%M:%S' , os.time()))
```

****using this example for now

Also see

os.clock() (on page 1-98)
settime() (on page 1-103)
settimezone()

print()

This function generates a response message.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
print(value1)
print(value1, value2)
print(value1, ..., valueN)
```

value1	The first argument to output
value2	The second argument to output
valueN	The last argument to output
...	One or more values separated with commas

Details

TSP-enabled instruments do not have inherent query commands. Like other scripting environments, the `print()` command and other related `print()` commands generate output. The `print()` command creates one response message.

The output from multiple arguments is separated with a tab character.

Numbers are printed using the `format.asciiprecision` attribute. If you want use Lua formatting, print the return value from the `tostring()` function.

Example 1

```
x = 10
print(x)
```

Example of an output response message:
`10`
Note that your output might be different if you set your ASCII precision setting to a different value.

Example 2

```
x = true
print(tostring(x))
```

Example of an output response message:
`true`

Also see

printbuffer()

This function prints data from tables or reading buffer subtables.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
printbuffer(startIndex, endIndex, bufferVar)
printbuffer(startIndex, endIndex, bufferVar, bufferVar2)
printbuffer(startIndex, endIndex, bufferVar, ..., bufferVarN)
```

startIndex	Beginning index of the buffer to print; this must be more than one and less than *endIndex*
endIndex	Ending index of the buffer to print; this must be more than *startIndex* and less than the index of the last entry in the tables
bufferVar	First table or reading buffer subtable to print
bufferVar2	Second table or reading buffer subtable to print
bufferVarN	The last table or reading buffer subtable to print
...	One or more tables or reading buffer subtables separated with commas

Details

If *startIndex* ≤ 1, 1 is used as *startIndex*. If *n* < *endIndex*, *n* is used as *endIndex*.

When any given reading buffers are used in overlapped commands that have not yet completed (at least to the specified index), this function outputs data as it becomes available.

When there are outstanding overlapped commands to acquire data, *n* refers to the index that the last entry in the table has after all the measurements have completed.

If you pass a reading buffer instead of a reading buffer subtable, the default subtable for that reading buffer is used.

This command generates a single response message that contains all data. The response message is stored in the output queue.

The `format.data` attribute controls the format of the response message.

Example

```
format.data = format.ASCII
format.asciiprecision = 6
printbuffer(1, rb1.n, rb1)
```

This assumes that `rb1` is a valid reading buffer in the runtime environment. The use of `rb1.n` (*bufferVar*`.n`) indicates that the instrument should output all readings in the reading buffer. In this example, `rb1.n` equals 10.

Example of output data (`rb1.readings`):

```
4.07205e-05, 4.10966e-05, 4.06867e-05, 4.08865e-05, 4.08220e-05, 4.08988e-05,
   4.08250e-05, 4.09741e-05, 4.07174e-05, 4.07881e-05
```

Also see

bufferVar.n (on page 1-4)

bufferVar.readings (on page 1-5)

format.asciiprecision

format.byteorder (on page 1-55)
format.data (on page 1-57)
printnumber() (on page 1-102)

printnumber()

This function prints numbers using the configured format.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
printnumber(value1)
printnumber(value1, value2)
printnumber(value1, ..., valueN)
```

value1	First value to print in the configured format
value2	Second value to print in the configured format
valueN	Last value to print in the configured format
...	One or more values separated with commas

Details

There are multiple ways to use this function, depending on how many numbers are to be printed.

This function prints the given numbers using the data format specified by `format.data` and `format.asciiprecision`.

Example

```
format.asciiprecision = 10
x = 2.54
printnumber(x)
format.asciiprecision = 3
printnumber(x, 2.54321, 3.1)
```

Configure the ASCII precision to 10 and set x to 2.54.
Read the value of x based on these settings.
Change the ASCII precision to 3.
View how the change affects the output of x and some numbers.
Output:

```
2.540000000e+00
2.54e+00, 2.54e+00, 3.10e+00
```

Also see

format.asciiprecision
format.byteorder (on page 1-55)
format.data (on page 1-57)
print() (on page 1-100)

settime()

This function sets the present time and the real time clock.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
settime(seconds)
settime(seconds, fractional)
```

seconds	The time in seconds since January 1, 1970, UTC
fractional	The number of fractional seconds

Details

This function sets the date and time of the instrument based on the time parameter (specified 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

```
systemTime = os.time({year = 2025,
month = 7,
day = 31,
hour = 14,
min = 25})
settime(systemTime)
```

Sets the mainframe system date to July 31, 2025 at 2:25 pm.

Also see

os.clock() (on page 1-98)
os.time() (on page 1-99)

settimezone()

This function sets the local time zone.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
settimezone(offset)
settimezone("offset", "dstOffset", "dstStart", "dstEnd")
```

offset	String representing offset from UTC
dstOffset	String representing the daylight savings offset from UTC
dstStart	String representing when daylight savings time starts
dstEnd	String representing when daylight savings time ends

Details

You only need to set the time zone if you use the `os.time()` and `os.date()` functions.

If only one parameter is given, the same time offset is used throughout the year. If four parameters are given, time is adjusted twice during the year for daylight savings time.

offset and *dstOffset* are strings of the form `"[+|-]hh[:mm[:ss]]"` that indicate how much time must be added to the local time to get UTC time:

- hh is a number between 0 and 23 that represents hours
- `mm` is a number between 0 and 59 that represents minutes
- `ss` is a number between 0 and 59 that represents seconds

The minute, second, +, and − fields are optional.

For example, to set the UTC-5 time zone, you specify the string `"5"`, because UTC-5 is 5 hours behind UTC and you must add 5 hours to the local time to determine UTC time. To specify the time zone UTC4, you specify `"-4"`, because UTC4 is 4 hours ahead of UTC and 4 hours must be subtracted from the local time to determine UTC.

dstStart and *dstEnd* are strings of the form `"MM.w.dw/hh[:mm[:ss]]"` that indicate when daylight savings time begins and ends respectively:

- `MM` is a number between 1 and 12 that represents the month
- `w` is a number between 1 and 5 that represents the week in the month
- `dw` is a number between 0 and 6 that represents the day of the week (where 0 is Sunday)

The rest of the fields represent the time of day that the change takes effect:

- hh represents hours
- `mm` represents minutes
- `ss` represents seconds

The minutes and seconds fields are optional.

The week of the month and day of the week fields are not specific dates.

Example

`settimezone("8", "1", "3.3.0/02", "11.2.0/02")`
`   settimezone(offset)`

Sets `offset` to equal +8 hours, +1 hour for DST, starts on Mar 14 at 2:00 am, ends on Nov 7 at 2:00 am. Sets local time zone to `offset`.

Also see

gettimezone() (on page 1-65)
settime()

os.time()

slot.autorestart

This attribute controls whether or not the instrument automatically restarts modules that have been powered-down.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Power cycle	Nonvolatile memory	localnode.DISABLE

Usage

```
autorestart = slot.autorestart
slot.autorestart = autorestart
```

`autorestart`	Automatic restart setting: ■ `localnode.ENABLE`: Automatically power-on and restart modules ■ `localnode.DISABLE`: Do not automatically power-on and restart modules

Details

This attribute controls whether modules that were powered down due to an abnormal condition (such as excessive power supply temperature) will restart when the condition is resolved.

Example

`slot.autorestart = localnode.ENABLE`
Enables automatic restart of modules that have been powered down due an abnormal condition.

Also see

slot.autostart (on page 1-106)
slot.start() (on page 1-107)
slot.stop() (on page 1-108)

slot.autostart

This attribute controls whether or not the instrument automatically starts modules that have been inserted.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Power cycle	Nonvolatile memory	localnode.ENABLE

Usage

```
autostart = slot.autostart
slot.autostart = autostart
```

autostart	Automatic start setting: ■ `localnode.ENABLE`: Automatically power on and start modules ■ `localnode.DISABLE`: Do not automatically power on and start modules

Details

The instrument will automatically power up and start modules that are inserted when the mainframe is powered-on as well as modules that are inserted when the instrument is running if this attribute is enabled. The instrument will never automatically power up a module that was stopped by the user.

Example

`slot.autostart = localnode.ENABLE`
Enables automatic start of modules that have been inserted

Also see

slot.autorestart (on page 1-105)
slot.start (on page 1-107)
slot.stop (on page 1-108)

slot.start()

This function powers-on a module in a specified slot.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
slot.start(slot)
```

slot	Module slot number

Details

This function will power-on the module in the specified slot. This command will not wait for the module to finish starting. Use `waitcomplete()` to wait for the module to finish starting.

Example

`slot.start(1)`
Powers-on the module installed in slot 1.

Also see

slot.autorestart (on page 1-105)
slot.autostart (on page 1-106)
slot.stop() (on page 1-108)

slot.stop()

This function powers-down a module in a specified slot.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
slot.stop(slot)
```

slot	Module slot number

Details

This function will power-down the module in the specified slot. This command can be used to prepare a module for safe removal while keeping the mainframe powered-on.

This command will not wait for the module to finish powering-down. Use `waitcomplete()` to wait for the module to finish powering-down.

Example

`slot.stop(1)`
Powers-down the module installed in slot 1.

Also see

slot.autorestart (on page 1-105)
slot.autostart (on page 1-106)
slot.start() (on page 1-107)

trigger.clear()

This function clears the command interface trigger event detector.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
trigger.clear()
```

Details

The trigger event detector indicates if a trigger event has been detected since the last `trigger.wait()` call. `trigger.clear()` clears the trigger event detector and discards the history of command interface trigger events.

Also see

trigger.wait() (on page 1-111)

trigger.EVENT_ID

This constant contains the command interface trigger event number.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Constant	Yes			

Usage

```
eventID = trigger.EVENT_ID
```

eventID	The event ID for the command interface triggers

Details

You can set the stimulus of any trigger object to the value of this constant to have the trigger object respond to command interface trigger events.

Example

```
trigger.timer[1].stimulus = trigger.EVENT_ID
```

Sets the trigger stimulus of trigger timer 1 to the command interface trigger event.

Also see

None

trigger.generator[N].assert()

This function generates a trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.generator[N].assert()
```

N	The generator number (1 to 8)

Details

Use this function to directly trigger events from the command interface or a script. For example, you can trigger a sweep while the instrument is under script control.

Example

```
trigger.generator[2].assert()
```

Generates a trigger event on generator 2

Also see

trigger.generator[N].EVENT_ID (on page 1-110)

trigger.generator[N].EVENT_ID

This constant identifies the trigger event generated by the trigger event generator.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Constant	Yes			

Usage

```
eventID = trigger.generator[N].EVENT_ID
```

eventID	The trigger event number
N	The generator number (1 to 8)

Details

This constant is an identification number that identifies events generated by this generator.

To have another trigger object respond to trigger events generated by this generator, set the other object's stimulus attribute to the value of this constant.

Example

`digio.trigger[5].stimulus = trigger.generator[2].EVENT_ID`
Uses a trigger event on generator 2 to be the stimulus for digital I/O trigger line 5.

Also see

trigger.generator[N].assert() (on page 1-109)

trigger.wait()

This function waits for a command interface trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
triggered = trigger.wait(timeout)
```

triggered	`true`: A trigger was detected during the timeout period `false`: No triggers were detected during the timeout period
timeout	Maximum amount of time in seconds to wait for the trigger

Details

This function waits up to *`timeout`* seconds for a trigger on the active command interface. A command interface trigger occurs when:

- A GPIB GET command is detected (GPIB only)
- A GPIB GET command is detected (GPIB only)
- A USBTMC TRIGGER message is received (USB only)
- A VXI-11 device_trigger method is invoked (VXI-11 only)
- A `*TRG` message is received

If one or more of these trigger events were previously detected, this function returns immediately.

After waiting for a trigger with this function, the event detector is automatically reset and rearmed. This is true regardless of the number of events detected.

Example

```
triggered = trigger.wait(10)
print(triggered)
```

Waits up to 10 seconds for a trigger.
If `false` is returned, no trigger was detected during the 10-second timeout.
If `true` is returned, a trigger was detected.

Also see

trigger.clear() (on page 1-108)

tsplink.group

This attribute contains the group number of a TSP-Link node.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Power cycle	Not applicable	0

Usage

```
groupNumber = tsplink.group
tsplink.group = groupNumber
```

groupNumber	The group number of the TSP-Link node: 0 to 64

Details

To remove the node from all groups, set the attribute value to 0.

When the node is turned off, the group number for that node changes to 0.

The master node can be assigned to any group. You can also include other nodes in the group that includes the master. Note that any nodes that are set to 0 are automatically included in the group that contains the master node, regardless of the group that is assigned to the master node.

Example

```
tsplink.group = 3
```

Assign the instrument to TSP-Link group number 3.

Also see

tsplink.master

This attribute reads the node number assigned to the master node.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
masterNodeNumber = tsplink.master
```

masterNodeNumber	The node number of the master node

Details

After doing a TSP-Link reset (`tsplink.reset()`), use this attribute to access the node number of the master in a set of instruments connected over TSP-Link.

Example

```
LinkMaster = tsplink.master
```

Store the TSP-Link master node number in a variable called LinkMaster.

Also see

tsplink.reset() (on page 1-116)

tsplink.node

This attribute defines the node number.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Not applicable	Nonvolatile memory	1

Usage

```
nodeNumber = tsplink.node
tsplink.node = nodeNumber
```

nodeNumber	The node number of the instrument or enclosure: 1 to 64

Details

This command sets the TSP-Link node number and saves the value in nonvolatile memory.

Changes to the node number do not take effect until `tsplink.reset()` from an earlier TSP-Link instrument is executed on any node in the system.

Each node connected to the TSP-Link system must be assigned a different node number.

Example

`tsplink.node = 3`
Sets the TSP-Link node for this instrument to number 3.

Also see

tsplink.reset() (on page 1-116)
tsplink.state (on page 1-117)

tsplink.readbit()

This function reads the state of a TSP-Link synchronization line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
data = tsplink.readbit(N)
```

data	The state of the synchronization line
N	The trigger line: 1 to 3

Details

Returns a value of zero (0) if the line is low and 1 if the line is high.

Example

```
data = tsplink.readbit(3)
print(data)
```

Assume line 3 is set high, and it is then read.
Output:
```
1.000000e+00
```

Also see

tsplink.readport() (on page 1-115)
tsplink.writebit() (on page 1-130)

tsplink.readport()

This function reads the TSP-Link trigger lines as a digital I/O port.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`data = tsplink.readport()`

data	Numeric value that indicates which lines are set

Details

The binary equivalent of the returned value indicates the input pattern on the I/O port. The least significant bit of the binary number corresponds to line 1 and the value of bit 3 corresponds to line 3. For example, a returned value of 2 has a binary equivalent of 010. This indicates that line 2 is high (1), and that the other two lines are low (0).

Example

```
data = tsplink.readport()
print(data)
```

Reads state of all three TSP-Link lines.
Assuming line 2 is set high, the output is:
```
2.000000e+00
```
(binary 010)
The format of the output may vary depending on the ASCII precision setting.

Also see

TSP-Link trigger lines
tsplink.readbit() (on page 1-114)
tsplink.writebit() (on page 1-130)
tsplink.writeport() (on page 1-131)

tsplink.reset()

This function initializes (resets) all nodes (instruments) in the TSP-Link system.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
nodesFound = tsplink.reset()
nodesFound = tsplink.reset(expectedNodes)
```

nodesFound	The number of nodes actually found on the system
expectedNodes	The number of nodes expected on the system (1 to 64)

Details

This function erases all information regarding other nodes connected on the TSP-Link system and regenerates the system configuration. This function must be called at least once before any remote nodes can be accessed. If the node number for any instrument is changed, the TSP-Link must be reset again.

If *expectedNodes* is not given, this function generates an error if no other nodes are found on the TSP-Link network.

If *nodesFound* is less than *expectedNodes*, an error is generated. Note that the node on which the command is running is counted as a node. For example, giving an expected node count of 1 does not generate any errors, even if there are no other nodes on the TSP-Link network.

Also returns the number of nodes found.

Example

```
nodesFound = tsplink.reset(2)
print("Nodes found = " .. nodesFound)
```

Perform a TSP-Link reset and indicate how many nodes are found.
Sample output if two nodes are found:

```
Nodes found = 2
```

Sample output if fewer nodes are found and if `localnode.showerrors = 1`:

```
1219, TSP-Link found fewer nodes than expected
Nodes found = 1
```

Also see

localnode.showerrors
tsplink.node (on page 1-113)
tsplink.state (on page 1-117)

tsplink.state

This attribute describes the TSP-Link online state.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
state = tsplink.state
```

state	TSP-Link state: `online` or `offline`

Details

When the instrument power is first turned on, the state is `offline`. After `tsplink.reset()` is successful, the state is `online`.

Example

`state = tsplink.state`
`print(state)`

Read the state of the TSP-Link system. If it is online, the output is: `online`

Also see

tsplink.node (on page 1-113)

tsplink.reset() (on page 1-116)

tsplink.trigger[N].assert()

This function simulates the occurrence of the trigger and generates the corresponding event ID.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
tsplink.trigger[N].assert()
```

N	The trigger line: 1 to 3

Details

The set pulse width determines how long the trigger is asserted.

Example

`tsplink.trigger[2].assert()`
Asserts trigger on trigger line 2.

Also see

tsplink.trigger[N].clear() (on page 1-119)
tsplink.trigger[N].mode (on page 1-121)
tsplink.trigger[N].overrun (on page 1-123)
tsplink.trigger[N].pulsewidth (on page 1-124)
tsplink.trigger[N].release() (on page 1-125)
tsplink.trigger[N].stimulus (on page 1-127)
tsplink.trigger[N].wait() (on page 1-129)

tsplink.trigger[N].clear()

This function clears the event detector for a LAN trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`tsplink.trigger[N].clear()`

N	The trigger line: 1 to 3

Details

The trigger event detector enters the detected state when an event is detected. `tsplink.trigger[N].clear()` clears a trigger event detector, discards the history of the trigger line, and clears the `tsplink.trigger[N].overrun` attribute.

Example

`tsplink.trigger[2].clear()`
Clears trigger event on synchronization line 2.

Also see

tsplink.trigger[N].mode (on page 1-121)
tsplink.trigger[N].overrun (on page 1-123)
tsplink.trigger[N].release() (on page 1-125)
tsplink.trigger[N].stimulus (on page 1-127)
tsplink.trigger[N].wait() (on page 1-129)

tsplink.trigger[N].EVENT_ID

This constant identifies the number that is used for the trigger events.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Constant	Yes			

Usage

```
eventID = tsplink.trigger[N].EVENT_ID
```

eventID	The trigger event number
N	The trigger line: 1 to 3

Details

This number is used by the TSP-Link trigger line when it detects an input trigger.

Set the stimulus of any trigger object to the value of this constant to have the trigger object respond to trigger events from this line.

Example

```
trigger.timer[1].stimulus = tsplink.trigger[2].EVENT_ID
```

Sets the trigger stimulus of trigger timer 1 to the TSP-Link trigger 2 event.

Also see

None

tsplink.trigger[N].mode

This attribute defines the trigger operation and detection mode.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Instrument reset Recall setup TSP-Link trigger N reset	Not saved	0 (tsplink.TRIG_BYPASS)

Usage

```
mode = tsplink.trigger[N].mode
tsplink.trigger[N].mode = mode
```

mode	The trigger mode
N	The trigger line: 1 to 3

Details

This attribute controls the mode in which the trigger event detector and the output trigger generator operate on the given trigger line.

The setting for the *mode* parameter can be one of the values shown in the following table.

Mode	Number value	Description
`tsplink.TRIG_BYPASS`	0	Allows direct control of the line as a digital I/O line.
`tsplink.TRIG_FALLING`	1	Detects falling-edge triggers as input. Asserts a TTL-low pulse for output.
`tsplink.TRIG_RISING`	2	If the programmed state of the line is high, the `tsplink.TRIG_RISING` mode behaves similarly to `tsplink.TRIG_RISINGA`. If the programmed state of the line is low, the `tsplink.TRIG_RISING` mode behaves similarly to `tsplink.TRIG_RISINGM`. Use `tsplink.TRIG_RISINGA` if the line is in the high output state. Use `tsplink.TRIG_RISINGM` if the line is in the low output state.
`tsplink.TRIG_EITHER`	3	Detects rising- or falling-edge triggers as input. Asserts a TTL-low pulse for output.
`tsplink.TRIG_SYNCHRONOUSA`	4	Detects the falling-edge input triggers and automatically latches and drives the trigger line low.
`tsplink.TRIG_SYNCHRONOUS`	5	Detects the falling-edge input triggers and automatically latches and drives the trigger line low. Asserts a TTL-low pulse as an output trigger.
`tsplink.TRIG_SYNCHRONOUSM`	6	Detects rising-edge triggers as an input. Asserts a TTL-low pulse for output.
`tsplink.TRIG_RISINGA`	7	Detects rising-edge triggers as input. Asserts a TTL-low pulse for output.
`tsplink.TRIG_RISINGM`	8	Edge detection as an input is not available. Generates a TTL-high pulse as an output trigger.

When programmed to any mode except `tsplink.TRIG_BYPASS`, the output state of the I/O line is controlled by the trigger logic and the user-specified output state of the line is ignored.

When the trigger mode is set to `tsplink.TRIG_RISING`, the user-specified output state of the line is examined. If the output state selected when the mode is changed is high, the actual mode that is used is `tsplink.TRIG_RISINGA`. If the output state selected when the mode is changed is low, the actual mode that is used is `tsplink.TRIG_RISINGM`.

The *`mode`* parameter stores the trigger mode as a numeric value when the attribute is read.

To control the line state, use the `tsplink.TRIG_BYPASS` mode with the `tsplink.writebit()` and the `tsplink.writeport()` commands.

Example

```
tsplink.trigger[3].mode = tsplink.TRIG_RISINGM
```

Sets the trigger mode for synchronization line 3 to `tsplink.TRIG_RISINGM`.

Also see

digio.writebit()
digio.writeport()
tsplink.trigger[N].assert() (on page 1-118)
tsplink.trigger[N].clear() (on page 1-119)
tsplink.trigger[N].overrun (on page 1-123)
tsplink.trigger[N].release() (on page 1-125)
tsplink.trigger[N].reset() (on page 1-126)
tsplink.trigger[N].stimulus (on page 1-127)
tsplink.trigger[N].wait() (on page 1-129)

tsplink.trigger[N].overrun

This attribute indicates if the event detector ignored an event while in the detected state.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Instrument reset Recall setup TSP-Link trigger N clear TSP-Link trigger N reset	Not applicable	Not applicable

Usage

```
overrun = tsplink.trigger[N].overrun
```

overrun	Trigger overrun state
N	The trigger line: 1 to 3

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.

Example

`print(tsplink.trigger[1].overrun)`
If an event was ignored, displays `true`; if an event was not ignored, displays `false`.

Also see

tsplink.trigger[N].assert() (on page 1-118)
tsplink.trigger[N].clear() (on page 1-119)
tsplink.trigger[N].mode (on page 1-121)
tsplink.trigger[N].release() (on page 1-125)
tsplink.trigger[N].reset() (on page 1-126)
tsplink.trigger[N].stimulus (on page 1-127)
tsplink.trigger[N].wait() (on page 1-129)

tsplink.trigger[N].pulsewidth

This attribute sets the length of time that the trigger line is asserted for output triggers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Instrument reset TSP-Link trigger N reset Recall setup	Not saved	10e-6 (10 µs)

Usage

```
width = tsplink.trigger[N].pulsewidth
tsplink.trigger[N].pulsewidth = width
```

width	The pulse width in seconds
N	The trigger line: 1 to 3

Details

Setting the pulse width to 0 (seconds) asserts the trigger indefinitely.

Example

`tsplink.trigger[3].pulsewidth = 20e-6`
Sets pulse width for trigger line 3 to 20 µs.

Also see

tsplink.trigger[N].release() (on page 1-125)

tsplink.trigger[N].release()

This function releases a latched trigger on the given TSP-Link trigger line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
tsplink.trigger[N].release()
```

N	The trigger line: 1 to 3

Details

Releases a trigger that was asserted with an indefinite pulse width. It also releases a trigger that was latched in response to receiving a synchronous mode trigger.

Example

`tsplink.trigger[3].release()`
Releases trigger line 3.

Also see

tsplink.trigger[N].assert() (on page 1-118)
tsplink.trigger[N].clear() (on page 1-119)
tsplink.trigger[N].mode (on page 1-121)
tsplink.trigger[N].overrun (on page 1-123)
tsplink.trigger[N].pulsewidth (on page 1-124)
tsplink.trigger[N].stimulus (on page 1-127)
tsplink.trigger[N].wait() (on page 1-129)

tsplink.trigger[N].reset()

This function resets some of the TSP-Link trigger attributes to their factory defaults.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
tsplink.trigger[N].reset()
```

N	The trigger line: 1 to 3

Details

The `tsplink.trigger[N].reset()` function resets the following attributes to their factory defaults:

- `tsplink.trigger[N].mode`
- `tsplink.trigger[N].stimulus`
- `tsplink.trigger[N].pulsewidth`

This also clears `tsplink.trigger[N].overrun`.

Example

```
tsplink.trigger[3].reset()
```

Resets TSP-Link trigger line 3 attributes back to factory default values.

Also see

tsplink.trigger[N].mode (on page 1-121)
tsplink.trigger[N].overrun (on page 1-123)
tsplink.trigger[N].pulsewidth (on page 1-124)
tsplink.trigger[N].stimulus (on page 1-127)

tsplink.trigger[N].stimulus

This attribute specifies the event that causes the synchronization line to assert a trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Instrument reset Recall setup TSP-Link trigger N reset	Not saved	0

Usage

```
eventID = tsplink.trigger[N].stimulus
tsplink.trigger[N].stimulus = eventID
```

eventID	The event identifier for the triggering event
N	The trigger line (1 to 3)

Details

To disable automatic trigger assertion on the synchronization line, set this attribute to zero (0).

Do not use this attribute when triggering under script control. Use `tsplink.trigger[N].assert()` instead.

The *eventID* parameter may be one of the existing trigger event IDs shown in the following table.

*******These details need work. We cannot use specific event IDs for Trebuchet.***************

Trigger event IDs*

Event ID	Event description
`***Set variable***.trigger.SWEEPING_EVENT_ID`	Occurs when the source-measure unit (SMU) transitions from the idle state to the arm layer of the trigger model
`***Set variable***.trigger.ARMED_EVENT_ID`	Occurs when the SMU moves from the arm layer to the trigger layer of the trigger model
`***Set variable***.trigger.SOURCE_COMPLETE_EVENT_ID`	Occurs when the SMU completes a source action
`***Set variable***.trigger.MEASURE_COMPLETE_EVENT_ID`	Occurs when the SMU completes a measurement action
`***Set variable***.trigger.PULSE_COMPLETE_EVENT_ID`	Occurs when the SMU completes a pulse
`***Set variable***.trigger.SWEEP_COMPLETE_EVENT_ID`	Occurs when the SMU completes a sweep
`***Set variable***.trigger.IDLE_EVENT_ID`	Occurs when the SMU returns to the idle state
`digio.trigger[N].EVENT_ID`	Occurs when an edge is detected on a digital I/O line
`tsplink.trigger[N].EVENT_ID`	Occurs when an edge is detected on a TSP-Link line
`lan.trigger[N].EVENT_ID`	Occurs when the appropriate LXI trigger packet is received on LAN trigger object *N*
`display.trigger.EVENT_ID`	Occurs when the TRIG key on the front panel is pressed

Trigger event IDs*	
Event ID	**Event description**
`trigger.EVENT_ID`	Occurs when a `*TRG` command is received on the remote interface GPIB only: Occurs when a `GET` bus command is received USB only: Occurs when a USBTMC TRIGGER message is received VXI-11 only: Occurs with the VXI-11 command `device_trigger`; reference the VXI-11 standard for additional details on the device trigger operation
`trigger.blender[N].EVENT_ID`	Occurs after a collection of events is detected
`trigger.timer[N].EVENT_ID`	Occurs when a delay expires
`trigger.generator[N].EVENT_ID`	Occurs when the `trigger.generator[N].assert()` function is executed
* Use the name of the trigger event ID to set the stimulus value rather than the numeric value. Using the name makes the code compatible for future upgrades (for example, if the numeric values must change when enhancements are added to the instrument).	

`print(tsplink.trigger[3].stimulus)`	Prints the event that starts TSP-Link trigger line 3 action.

Also see

tsplink.trigger[N].assert() (on page 1-118)
tsplink.trigger[N].reset() (on page 1-126)

tsplink.trigger[N].wait()

This function waits for a trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
triggered = tsplink.trigger[N].wait(timeout)
```

triggered	Trigger detection indication; set to one of the following values: ■ A trigger is detected during the timeout period: `true` ■ A trigger is not detected during the timeout period: `false`
N	The trigger line: 1 to 3
timeout	The timeout value in seconds

Details

This function waits up to the timeout value for an input trigger. If one or more trigger events were detected since the last time `tsplink.trigger[N].wait()` or `tsplink.trigger[N].clear()` was called, this function returns immediately.

After waiting for a trigger with this function, the event detector is automatically reset and rearmed. This is true regardless of the number of events detected.

Example

```
triggered = tsplink.trigger[3].wait(10)
print(triggered)
```

Waits up to 10 seconds for a trigger on TSP-Link line 3.
If `false` is returned, no trigger was detected during the 10-second timeout.
If `true` is returned, a trigger was detected.

Also see

tsplink.trigger[N].clear() (on page 1-119)

tsplink.writebit()

This function sets a TSP-Link trigger line high or low.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`tsplink.writebit(N, data)`

N	The trigger line: 1 to 3
data	The value to write to the bit: ■ Low: `0` ■ High: `1`

Details

Use `tsplink.writebit()` and `tsplink.writeport()` to control the output state of the trigger line when trigger operation is set to `tsplink.TRIG_BYPASS`.

If the output line is write-protected by the `tsplink.writeprotect` attribute, this command is ignored.

The reset function does not affect the present states of the TSP-Link trigger lines.

Example

`tsplink.writebit(3, 0)`
Sets trigger line 3 low (0).

Also see

tsplink.readbit() (on page 1-114)
tsplink.readport() (on page 1-115)
tsplink.writeport() (on page 1-131)
tsplink.writeprotect (on page 1-131)

tsplink.writeport()

This function writes to all TSP-Link synchronization lines.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
tsplink.writeport(data)
```

data	Value to write to the port: 0 to 7

Details

The binary representation of `data` indicates the output pattern that is written to the I/O port. For example, a data value of 2 has a binary equivalent of 010. Line 2 is set high (1), and the other two lines are set low (0).

Write-protected lines are not changed.

Use the `tsplink.writebit()` and `tsplink.writeport()` commands to control the output state of the synchronization line when trigger operation is set to `tsplink.TRIG_BYPASS`.

The `reset()` function does not affect the present states of the trigger lines.

Example

`tsplink.writeport(3)`
Sets the synchronization lines 1 and 2 high (binary 011).

Also see

tsplink.writeprotect

This attribute contains the write-protect mask that protects bits from changes by the `tsplink.writebit()` and `tsplink.writeport()` functions.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Instrument reset Recall setup	Not saved	0

Usage

```
mask = tsplink.writeprotect
tsplink.writeprotect = mask
```

mask	An integer that specifies the value of the bit pattern for write-protect; set bits to 1 to write-protect the corresponding TSP-Link trigger line

Details

The binary equivalent of *`mask`* indicates the mask to be set for the TSP-Link trigger line. For example, a *`mask`* value of 5 has a binary equivalent of 101. This *`mask`* write-protects TSP-Link trigger lines 1 and 3.

Example

`tsplink.writeprotect = 5`
Write-protects TSP-Link trigger lines 1 and 3.

Also see

Controlling digital I/O lines
tsplink.readbit() (on page 1-114)
tsplink.readport() (on page 1-115)
tsplink.writebit() (on page 1-130)
tsplink.writeport() (on page 1-131)

userstring.add()

This function adds a user-defind string to nonvolatile memory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
userstring.add("name", "value")
```

name	The name of the string; the key of the key-value pair
value	The string to associate with *name*; the value of the key-value pair

Details

This function associates the string `value` with the string `name` and stores this key-value pair in nonvolatile memory.

Use the `userstring.get()` function to retrieve the `value` associated with the specified `name`.

You can use the `userstring` functions to store custom, instrument-specific information in the instrument, such as department number, asset number, or manufacturing plant location.

Example

```
userstring.add("assetnumber", "236")
userstring.add("product", "Widgets")
userstring.add("contact", "John Doe")
for name in userstring.catalog() do
   print(name .. " = " ..
      userstring.get(name))
end
```

Stores user-defined strings in nonvolatile memory and recalls them from the instrument using a for loop.
Example output:

```
assetnumber = 236
contact = John Doe
product = Widgets
```

Also see

userstring.delete() (on page 1-134)
userstring.get() (on page 1-134)
userstring.table() (on page 1-136)

userstring.delete()

This function deletes a user-defined string from nonvolatile memory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
userstring.delete("name")
```

name	The name (key) of the key-value pair of the user-defined string to delete

Details

This function deletes the string that is associated with *name* from nonvolatile memory.

Example

```
userstring.delete("assetnumber")
userstring.delete("product")
userstring.delete("contact")
```

Deletes the user-defined strings associated with the `assetnumber`, `product`, and `contact` names.

Also see

userstring.add() (on page 1-133)
userstring.get() (on page 1-134)
userstring.table() (on page 1-136)

userstring.get()

This function retrieves a user-defined string from nonvolatile memory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
value = userstring.get("name")
```

value	The value of the user-defined string key-value pair
name	The name (key) of the user-defined string

Details

This function retrieves the string that is associated with *name* from nonvolatile memory.

Example

```
userstring.add("assetnumber", "236")
value = userstring.get("assetnumber")
print(value)
```

Create the user-defined string `assetnumber`, set to a value of `236`.
Read the value associated with the user-defined string named `assetnumber`.
Store it in a variable called `value`, then print the variable `value`.
Output:

```
236
```

Also see

userstring.add() (on page 1-133)
userstring.delete() (on page 1-134)
userstring.table() (on page 1-136)

userstring.table()

This function creates an iterator for the user-defined string catalog.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
userstringtable = userstring.table()
for name in userstring.table() do body end
```

userstringtable	A table that holds copies of all user string pairs
name	The name of the string; the key of the key-value pair
body	Code to execute in the body of the for loop

Details

The catalog provides access for user-defined string pairs, allowing you to manipulate all the key-value pairs in nonvolatile memory. The entries are enumerated in no particular order.

Example 1

```
for name in userstring.table() do
   userstring.delete(name)
end
```

Deletes all user-defined strings in nonvolatile memory.

Example 2

```
userstring.add("assetnumber", "236")
userstring.add("product", "Widgets")
userstring.add("contact", "John Doe")
for name, value in pairs(userstring.table()) do
   print(name .. " = " ..
      "value")
end
```

Prints all `userstring` key-value pairs.

Output:

```
product = Widgets
assetnumber = 236
contact = John Doe
```

Notice the key-value pairs are not listed in the order they were added.

Also see

userstring.add() (on page 1-133)
userstring.delete() (on page 1-134)
userstring.get() (on page 1-134)