Model DMM7510
7½ Digit Graphical Sampling Multimeter

Reference Manual

A Greater Measure of Confidence

KEITHLEY
A Tektronix Company

Model DMM7510

7½ Digit Graphical Sampling Multimeter

Reference Manual

KEITHLEY
A Tektronix Company

The following safety precautions should be observed before using this product and any associated instrumentation. Although some instruments and accessories would normally be used with nonhazardous voltages, there are situations where hazardous conditions may be present.

This product is intended for use by qualified personnel who recognize shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product. Refer to the user documentation for complete product specifications.

If the product is used in a manner not specified, the protection provided by the product warranty may be impaired.

The types of product users are:

Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring that operators are adequately trained.

Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument. They must be protected from electric shock and contact with hazardous live circuits.

Maintenance personnel perform routine procedures on the product to keep it operating properly, for example, setting the line voltage or replacing consumable materials. Maintenance procedures are described in the user documentation. The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.

Service personnel are trained to work on live circuits, perform safe installations, and repair products. Only properly trained service personnel may perform installation and service procedures.

Keithley Instruments products are designed for use with electrical signals that are measurement, control, and data I/O connections, with low transient overvoltages, and must not be directly connected to mains voltage or to voltage sources with high transient overvoltages. Measurement Category II (as referenced in IEC 60664) connections require protection for high transient overvoltages often associated with local AC mains connections. Certain Keithley measuring instruments may be connected to mains. These instruments will be marked as category II or higher.

Unless explicitly allowed in the specifications, operating manual, and instrument labels, do not connect any instrument to mains.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30 V RMS, 42.4 V peak, or 60 VDC are present. A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000 V, no conductive part of the circuit may be exposed.

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance-limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, ensure that the line cord is connected to a properly-grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.

For safety, instruments and accessories must be used in accordance with the operating instructions. If the instruments or accessories are used in a manner not specified in the operating instructions, the protection provided by the equipment may be impaired.

Do not exceed the maximum signal levels of the instruments and accessories, as defined in the specifications and operating information, and as shown on the instrument or test fixture panels, or switching card.

When fuses are used in a product, replace with the same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as protective earth (safety ground) connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

If a ⏚ screw is present, connect it to protective earth (safety ground) using the wire recommended in the user documentation.

The ⚠ symbol on an instrument means caution, risk of danger. The user must refer to the operating instructions located in the user documentation in all cases where the symbol is marked on the instrument.

The ⚡ symbol on an instrument means caution, risk of electric shock. Use standard safety precautions to avoid personal contact with these voltages.

The ♨ symbol on an instrument shows that the surface may be hot. Avoid personal contact to prevent burns.

The ⏚ symbol indicates a connection terminal to the equipment frame.

If this (Hg) symbol is on a product, it indicates that mercury is present in the display lamp. Please note that the lamp must be properly disposed of according to federal, state, and local laws.

The **WARNING** heading in the user documentation explains dangers that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading in the user documentation explains hazards that could damage the instrument. Such damage may invalidate the warranty.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits — including the power transformer, test leads, and input jacks — must be purchased from Keithley Instruments. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. Other components that are not safety-related may be purchased from other suppliers as long as they are equivalent to the original component (note that selected parts should be purchased only through Keithley Instruments to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call a Keithley Instruments office for information.

To clean an instrument, use a damp cloth or mild, water-based cleaner. Clean the exterior of the instrument only. Do not apply cleaner directly to the instrument or allow liquids to enter or spill on the instrument. Products that consist of a circuit board with no case or chassis (e.g., a data acquisition board for installation into a computer) should never require cleaning if handled according to instructions. If the board becomes contaminated and operation is affected, the board should be returned to the factory for proper cleaning/servicing.

Safety precaution revision as of January 2013.

Table of Contents

Introduction

Welcome

Thank you for choosing a Keithley Instruments product. The Model DMM7510 is a 7½ digit graphical sampling multimeter that expands standard DMM functions with high-speed digitizing and large graphical color touchscreen display. This DMM offers a broad range of measurement capabilities, including 17 measurement functions. In addition to industry-leading DC accuracies, functions such as capacitance, 10 amp current, and 18-bit current and voltage digitizing are included. Tying all these features together is a large 5-inch color touchscreen display that brings users an unprecedented combination of data visualization and interaction, enabling users to gain deeper insight into their measurements.

The Model DMM7510 provides superior measurement accuracy and the speed necessary for a broad range of applications, from system applications and production testing to benchtop applications. The Model DMM7510 meets application requirements for production engineers, research and development engineers, test engineers, and scientists.

Extended warranty

Additional years of warranty coverage are available on many products. These valuable contracts protect you from unbudgeted service expenses and provide additional years of protection at a fraction of the price of a repair. Extended warranties are available on new and existing products. Contact your local Keithley Instruments office, sales partner, or distributor for details.

Contact information

If you have any questions after you review the information in this documentation, please contact your local Keithley Instruments office, sales partner, or distributor, or call Keithley Instruments corporate headquarters (toll-free inside the U.S. and Canada only) at 1-800-935-5595, or from outside the U.S. at +1-440-248-0400. For worldwide contact numbers, visit the Keithley Instruments website (*http://www.keithley.com*).

CD-ROM contents

Each Model DMM7510 instrument is shipped with the 7½ Digit Multimeter Product Information CD-ROM (Keithley Instruments part number DMM7510-950-01).

The Model DMM7510 7½ Digit Multimeter Product Information CD-ROM contains:

- **Quick Start Guide:** Provides unpacking instructions, describes basic connections, reviews basic operation information, and provides a quick test procedure to ensure the instrument is operational.
- **User's Manual:** Provides application examples that you can use as a starting point to create your own applications.
- **Reference Manual:** Includes advanced operation topics, maintenance information, troubleshooting procedures, and in-depth descriptions of programming commands.
- **KickStart Startup Software Quick Start Guide:** Provides instructions for the KickStart Startup Software, which allows you to quickly make measurements and get results without having to program test scripts.
- **Accessories information:** Documentation for accessories that are available for the Model DMM7510.

For the latest drivers and additional support information, see the Keithley Instruments website (*http://www.keithley.com*).

Organization of manual sections

The information in this manual is organized into the following major categories:

- **General operation:** Describes the components of the instrument and basic operation.
- **Functions and features:** Describes features and functions, such as relative offset, filters, reading buffers, configuration lists, triggering, the digital I/O port, and TSP-Link synchronization lines.
- **Measure considerations:** Describes best practices and recommended procedures that can increase measurement speed, accuracy, and sensitivity.
- **Introduction to SCPI commands:** Describes how to control the instrument using SCPI commands.
- **SCPI command reference:** Contains programming notes and an alphabetical listing of all SCPI commands available for the Model DMM7510.
- **Introduction to TSP operation:** Describes the basics of using Test Script Processor (TSP®) commands to control the instrument and describes how to control the instrument using TSP commands and Test Script Builder (TSB®) software, TSP-Link system expansion, and TSP-Net.
- **TSP command reference:** Contains programming notes and an alphabetical listing of all TSP commands available for the Model DMM7510.
- **Frequently asked questions:** Contains information that answers commonly asked questions.

- **Next steps:** Contains sources of additional information.
- **Maintenance:** Contains information about instrument maintenance, including line fuse replacement and firmware upgrades.
- **Common commands:** Contains descriptions of IEEE Std. 488.2 common commands.
- **Status model:** Describes the Model DMM7510 status model.

The PDF version of this manual contains bookmarks for each section. The manual sections are also listed in the Table of Contents at the beginning of this manual.

For more information about bookmarks, see Adobe® Acrobat® or Reader® help.

Capabilities and features

The Model DMM7510 has the following features:

- High-resolution, five-inch touchscreen display with enhanced graphical data visualization and on-screen debug and error histories
- Ability to perform sensitive measurements on low-level signals
- Simplified trigger model with measure configuration lists
- Front-panel USB-A connector for flash-drive support; rear-panel USB-B connector for communication, control, and data transfer

Some additional capabilities of the Model DMM7510:

- Limit testing with a built-in comparator for pass/fail testing
- Digital I/O for stand-alone binning operations or interface to a component handler
- SCPI and Test Script Processor (TSP™) programming languages with remote interface ports (IEEE-488/GPIB, USB, and LAN)
- Built-in math expressions and user-defined expressions (using a remote interface)
- Filtering to reduce reading noise
- Trigger model supports extensive triggering and synchronization schemes at hardware speeds
- LXI® Core Specification 1.4 compliance
- TSP-Link® system expansion interface that test system builders can use to connect multiple instruments in a master and subordinate configuration.
- Supports IEEE-488 (GPIB), USB, and ethernet local area network (LAN) connections

General ratings

The Model DMM7510 instrument's general ratings and connections are listed in the following table.

Category	Specification
Supply voltage range	100 V_{RMS} to 240 V_{RMS}, 50 Hz or 60 Hz (autosensing at power on)
Input and output connections	See Rear panel overview (on page 2-6).
Environmental conditions	For indoor use only **Altitude:** Maximum 2000 meters (6562 feet) above sea level **Operating:** 0 °C to 50 °C (32 °F to 122 °F), ≤80 % relative humidity at 35 °C (95 °F) **Storage:** −30 °C to 70 °C (−22 °F to 158 °F) **Pollution degree:** 1 or 2

General operation

In this section:

Instrument power

Follow the steps below to connect the Model DMM7510 to line power and turn on the instrument. The Model DMM7510 operates from a line voltage of 100 V to 240 V at a frequency of 50 Hz or 60 Hz. It automatically senses line voltage and frequency. Make sure the operating voltage in your area is compatible.

You must turn on the Model DMM7510 and allow it to warm up for at least 90 minutes to achieve rated accuracies.

⚠ CAUTION

Operating the instrument on an incorrect line voltage may cause damage to the instrument, possibly voiding the warranty.

<div style="background-color:red; color:white; text-align:center;">

⚠ WARNING

</div>

The power cord supplied with the Model DMM7510 contains a separate protective earth (safety ground) wire for use with grounded outlets. When proper connections are made, the instrument chassis is connected to power-line ground through the ground wire in the power cord. In the event of a failure, not using a properly grounded protective earth and grounded outlet may result in personal injury or death due to electric shock.

Do not replace detachable mains supply cords with inadequately rated cords. Failure to use properly rated cords may result in personal injury or death due to electric shock.

Connect the power cord

To connect the power cord:

1. Make sure that the front panel POWER switch is in the off (O) position.
2. Connect the female end of the supplied power cord to the AC receptacle on the rear panel.
3. Connect the other end of the power cord to a grounded AC outlet.

Turn the Model DMM7510 on or off

To turn a Model DMM7510 on:

1. Disconnect any devices under test (DUTs) from the Model DMM7510.
2. Press the front-panel **POWER** switch to place it in the on (|) position.

The instrument displays a status bar as the instrument powers on. The home screen is displayed when power on is complete.

To turn a Model DMM7510 off:

1. Press the front-panel **POWER** switch to place it in the off (O) position.

Front panel overview

The front panel of the Model DMM7510 is shown below. Descriptions of the controls on the front panel follow the figure.

Figure 1: Model DMM7510 front panel

POWER switch		Turns the instrument on or off. To turn the instrument on, press the power switch so that it is in the on position (|). To turn it off, press the power switch so that it is in the off position (O).
HOME key		Returns the display to the home screen.
MENU key		Opens the main menu. Press the icons on the main menu to open measure, view, trigger, script, and system screens. For details, refer to Menu overview (on page 2-22).
QUICKSET key		Opens a menu of preconfigured setups, including Voltage Waveform, Interval Measure, Current Waveform, and External Scan. Also allows you to choose measure functions and adjust performance for better resolution or speed. Refer to Using Quick Setups (on page 2-132).
HELP key		Opens help for the area or item that is selected on the display. If there is no selection when you press the **HELP** key, it displays overview information for the screen you are viewing.

USB port		Saves reading buffer data and screen grabs to a USB flash drive. Also stores and retrieves scripts to and from a USB flash drive. The flash drive must be formatted as a FAT drive.
Touchscreen		The Model DMM7510 has a high-resolution, five-inch color touchscreen display. The touchscreen accesses swipe screens and menu options. You can access additional interactive screens by pressing the front-panel MENU, QUICKSET, and FUNCTION keys. Refer to Touchscreen display (on page 2-8) for details.
Navigation control		Moves the cursor and make screen selections. **Turning the navigation control:** Moves the cursor to highlight a list value or menu item so that you can select it. Turning the control when the cursor is in a value entry field increases or decreases the value in the field. **Pressing the navigation control:** Selects the highlighted choice or allows you to edit the selected field.
ENTER key		Selects the highlighted choice or allows you to edit the selected field.
EXIT key		Returns to the previous screen or closes a dialog box. For example, press the **EXIT** key when the main menu is displayed to return to the home screen. When you are viewing a subscreen (for example, the Event Log screen), press the **EXIT** key to return to the main menu screen.
FUNCTION key		Displays instrument functions. To select a function, touch the function name on the screen.
TRIGGER key		Accesses trigger-related settings and operations. The action of the TRIGGER key depends on the instrument state. For details, see Switching between measurement methods (on page 3-59).
REMOTE LED indicator	REMOTE ●	Illuminates when the instrument is controlled through a remote interface.
LAN LED indicator	LAN ●	Illuminates when the instrument is connected to a local area network (LAN).

1588 LED indicator	1588 🟢	Illuminates when the instrument is connected to an IEEE-1588 compliant device.

> ## NOTE
>
> 1588 functionality is not supported at this time. This functionality will be made available with a firmware update. See the *Model DMM7510 Release Notes* on the Keithley Instruments website (*http://www.keithley.com*) for details.
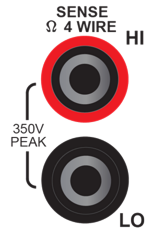
SENSE terminals	SENSE Ω 4 WIRE HI / 350V PEAK / LO	Use the SENSE HI and SENSE LO terminals and the INPUT terminals with the 4-wire resistance, 3-wire and 4-wire RTD temperature, and DC voltage ratio functions.
INPUT terminals	INPUT HI / 1000V PEAK / LO	Use the INPUT HI and INPUT LO terminals for all measurements except current.
FRONT/REAR TERMINALS switch	F R TERMINALS FRONT/REAR	Activates the terminals on the front or rear panel. When the front-panel terminals are active, a green "F" is visible to the left of the FRONT/REAR switch. When the rear-panel terminals are active, a yellow "R" is visible to the left of the switch.
AMPS	AMPS 3A, 250V	Use the AMPS connection with the INPUT LO terminal to measure $\leq$3A DC or AC$_{rms}$ current.
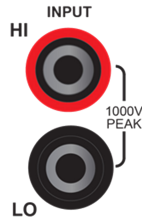
Rear panel overview

The rear panel of the Model DMM7510 is shown below. Descriptions of the options follow the figure.

Figure 2: Model DMM7510 rear panel
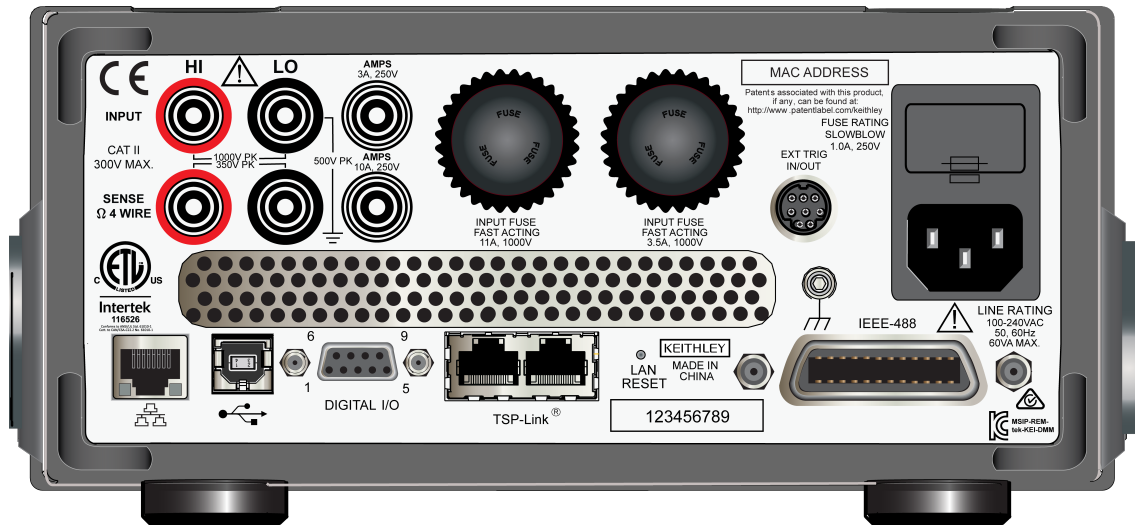
INPUT and SENSE terminals	Use the INPUT HI and INPUT LO terminals for all measurements except current. For current measurements, use the 3 A or 10 A AMPS connection with the INPUT LO terminal. Use the SENSE HI and SENSE LO terminals and the INPUT terminals with the 4-wire resistance, 3-wire and 4-wire RTD temperature, and DC voltage ratio functions.
AMPS connection	3 A, 250 V current connector for DC amp and digitize DC current 10 mA to 3 A ranges, and AC current 1 μA to 3 A ranges. 10 A, 250 V current connector for DC current and digitize DC current 10 A range and AC current 10 A range only.
Measurement Input Fuses	Fast-acting current-input fuses. For continued protection against fire hazard, replace these fuses with same type and rating. See Input fuse replacement (on page A-2) for details.
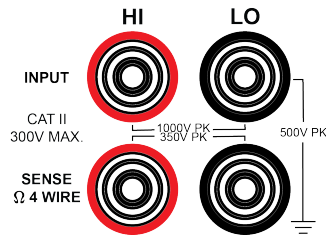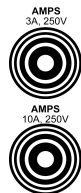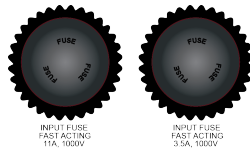
EXT TRIG IN/OUT terminal	EXT TRIG IN/OUT	This terminal is a TTL-compatible input/output line with a 0 to 5 V logic signal. You can use this line for triggering by using the transition of the line state to initiate an action. The instrument can generate output trigger pulses and detect input trigger pulses on this line. Refer to External I/O (on page 3-55) for details.
Line fuse and power receptacle		Connect the line cord to the power receptacle and a grounded AC power outlet. The line fuse, located just above the power receptacle, protects the power line input of the instrument. For safety precautions and other details, see Instrument power (on page 2-1) and Line fuse replacement (on page A-1).
LAN port		Supports full connectivity on a 10 Mbps or 100 Mbps network. The Model DMM7510 is an LXI version 1.4 Core 2011 compliant instrument that supports TCP/IP and complies with IEEE Std 802.3 (ethernet LAN). See LAN cable connection (on page 2-67).
USB port		USB-B connection for communication, control, and data transfer. For details, see USB communications (on page 2-75).
Digital I/O port	6 9 1 5	A digital input/output port that detects and outputs digital signals. The port provides six digital I/O lines. Each output is set high (+5 V) or low (0 V) and can read high or low logic levels. Each digital I/O line is an open-drain signal. Refer to Digital I/O (on page 3-43) for information.
TSP-Link ports		TSP-Link® system expansion interface, which test system builders can use to connect multiple instruments in a master and subordinate configuration. TSP-Link is a high-speed trigger synchronization and communication bus. For details, see TSP-Link System Expansion Interface (on page 3-102).
LAN reset	LAN RESET	Reverts the LAN settings and the instrument password to default values. See Reset LAN settings (on page 2-74) for more information.
Chassis ground		Ground screw for connections to chassis ground. This provides a connection terminal to the equipment frame.
IEEE-488 port		GPIB connection; the default setting for the Model DMM7510 is 16. Refer to GPIB setup (on page 2-63).
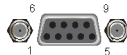
Touchscreen display

The touchscreen display gives you quick front-panel access to measure settings, system configuration, instrument and test status, reading buffer information, and other instrument functionality. The display has multiple swipe screens that you can access by swiping the front panel. You can access additional interactive screens by pressing the front-panel MENU, QUICKSET, and FUNCTION keys.

The following topics describe the features of the touchscreen in more detail.

⚠ CAUTION
Do not use sharp metal objects, such as tweezers or screwdrivers, or pointed objects, such as pens or pencils, to touch the touchscreen. It is strongly recommended that you use only fingers to operate the instrument. Use of clean-room gloves to operate the touchscreen is supported.

Select items on the touchscreen

To select an item on the displayed screen, do one of the following:

* Touch it with your finger
* Turn the navigation control to highlight the item, and then press the navigation control to select it

The following topics describe the Model DMM7510 touchscreen in more detail.

Scroll bars

Some of the interactive screens have additional options that are only visible when you scroll down the screen. A scroll indicator on the right side of the touchscreen identifies these screens. Swipe the screen up or down to view the additional options.

The figure below shows a screen with a scroll bar.
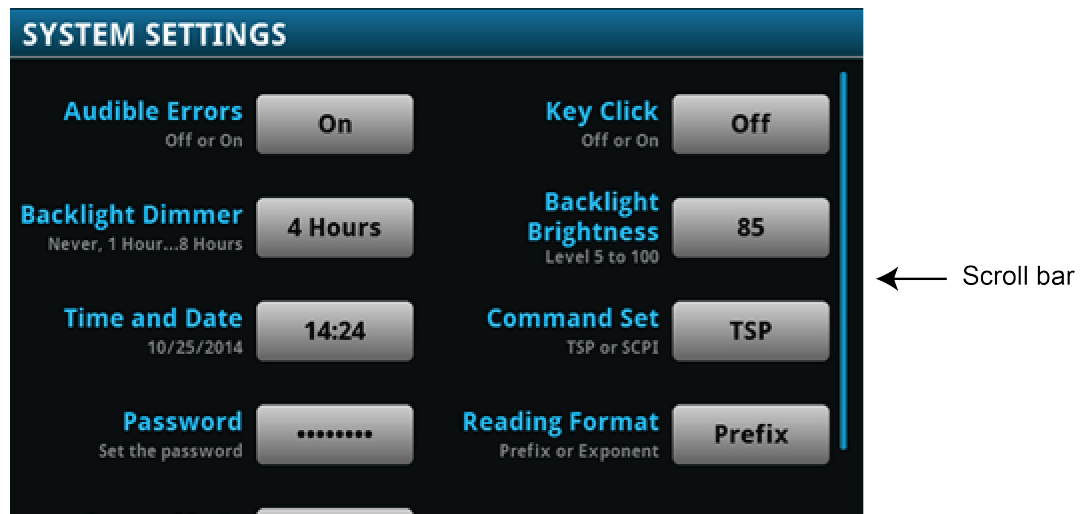
Enter information

Some of the menu options open a keypad or keyboard that you can use to enter information. For example, if you are setting the GPIB address from the front panel, you see the keypad shown in the following figure.

Figure 3: Front-panel keyboard for GPIB Address entry

You can enter information by touching the screen to select characters and options from the keypad or keyboard. You can move the cursor in the entry box by touching the screen. The cursor is moved to the spot in the entry box where you touched the screen. On number keypads, you can also use the navigation control to move the cursor to a specific number.

On number keypads, you can use the navigation control to set values:

1. Turn the control to underline the character that you want to change.
2. Press the control to select the character for edit.
3. Turn the control to scroll through the options.
4. Press the control to set the character.
5. Press the **ENTER** key to save the change.

On keyboards and keypads, you can use the navigation control to select characters.

Adjust the backlight brightness and dimmer

You can adjust the brightness of the Model DMM7510 touchscreen display and buttons from the front panel or over a remote interface. You can also set the backlight to dim after a specified period has passed with no front-panel activity (available from the front-panel display only). The backlight settings set through the front-panel display are saved through a reset or power cycle.

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

To adjust the backlight brightness from the front panel:

1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Select the button next to Backlight Brightness. The Backlight Brightness dialog box opens.
4. Drag the sliding adjustment to set the backlight.
5. Select **OK** to save your setting.

To set the backlight dimmer from the front panel:

1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Select the button next to Backlight Dimmer. The Backlight Dimmer dialog box opens.
4. Select a dimmer setting.

To adjust the brightness using the SCPI remote interface:

Send the following command:

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

Where `<brightness>` is one of the following options:

- Full brightness: `ON100`
- 75 % brightness: `ON75`
- 50 % brightness: `ON50`
- 25 % brightness: `ON25`
- Display off: `OFF`
- Display, key lights, and all indicators off: `BLACkout`

To adjust the backlight using TSP commands:

Send the following command:

```
display.lightstate = brightness
```

Where `brightness` is one of the following options:

- Full brightness: `display.STATE_LCD_100`

- 75 % brightness: `display.STATE_LCD_75`

- 50 % brightness: `display.STATE_LCD_50`

- 25 % brightness: `display.STATE_LCD_25`

- Display off: `display.STATE_LCD_OFF`

- Display, key lights, and all indicators off: `display.STATE_BLACKOUT`

Event messages

During operation and programming, front-panel messages may be briefly displayed. Messages are information, warning, or error notifications. For information on event messages, refer to .

Figure 4: Example front-panel error message
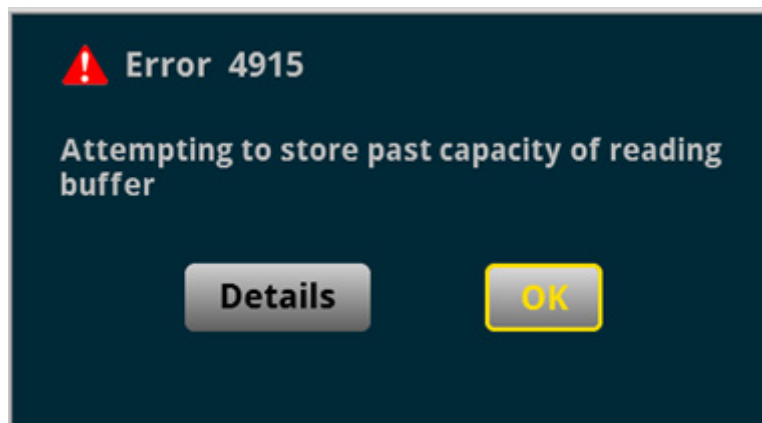
Screen descriptions

The following topics describe the screens and options that you can view on the Model DMM7510 front-panel display.

Home screen

This is the default screen that you see whenever you turn the Model DMM7510 on or when you press the HOME key. The options available on the home screen are described in the following topics.

Figure 5: Model DMM7510 home screen

Status and event indicators

The indicators at the top of the home screen contain information about instrument settings and states. Some of the indicators also provide access to instrument settings.

Press an indicator to get more information about the present state of the instrument. You can also select the indicators by turning the navigation control to select an indicator and then pressing **ENTER**.

Communications indicator

The communications indicator displays the type of communications the instrument is using. Press the indicator to display the present communications settings. Press the **Change Settings** button at the bottom of the dialog box to change the settings. Refer to Remote communications interfaces (on page 2-61) for detail on the options that are available.

Figure 6: Communications indicator expanded

Indicator	Instrument communication
Local	Instrument is controlled from the front panel
GPIB	Instrument is communicating through a GPIB interface
TCPIP	Instrument is communicating through a LAN interface
VXI-11	Instrument is communicating using VXI-11
USBTMC	Instrument is communicating through a USB interface
Telnet	Instrument is communicating through Telnet
TSP-Link	Instrument is communicating through TSP-Link
Slave	Instrument is a subordinate in a TSP-Link system

Communications activity indicator

The activity indicator is located to the right of the communications indicator. When the instrument is communicating with a remote interface, the up and down arrows flash.

Figure 7: Communications indicator

If a service request has been generated, SRQ is displayed to the right of the up and down arrows. You can instruct the instrument to generate a service request (SRQ) when one or more events or conditions occur. This indicator stays on until the serial poll byte is read or all the conditions that caused SRQ are cleared.

Active buffer indicator

The Reading Buffer indicator shows the name of the active reading buffer. Select the indicator to open a menu of available buffers. Select a buffer name in the list to make it the active reading buffer. The name of the new active reading buffer is updated in the indicator bar.

The green bar next to the buffer name indicates how full the buffer is.

Figure 8: Active buffer indicator menu
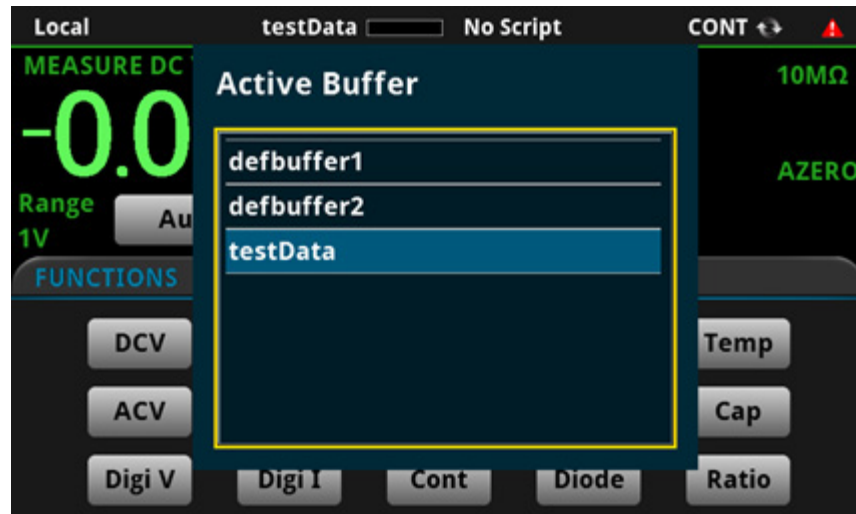
Active script indicator

This indicator shows script activity and allows you to control script action from the Home screen.

If there is no script activity, the indicator displays "No Script." You can select the indicator to display a menu of available scripts. Select a script name to run that script.

If a script is running from the instrument or the USB flash drive, the name of the script is displayed. If a script from TSB is running, "TSB_SCRIPT" is displayed. If you select the indicator, you are prompted to abort the running script.

If the instrument is recording a macro script, "Recording" is displayed. You can select the indicator to select an option to stop or cancel recording.

Figure 9: Model DMM7510 active script indicator
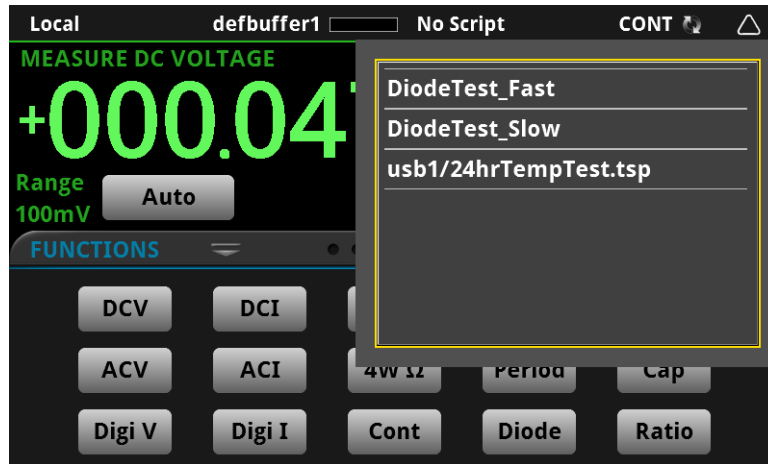
Trigger mode indicator

Located to the right of the active script indicator, this indicator shows the active trigger measurement method. Select the indicator to open a menu. Select one of the buttons on the menu to change the trigger measurement method or initiate or abort the trigger model. In the figure below, Continuous Measurement is the present trigger measurement method.

Figure 10: Trigger mode indicator

Indicator	Meaning
CONT	Continuous measurement: The instrument is making measurements continuously.
MAN	Manual trigger mode: Press the front-panel TRIGGER key to initiate a single measurement.
RUN	Trigger model measurement method. The instrument is running the presently selected trigger model.
IDLE	Trigger model measurement method. The trigger model is not running.
WAIT	Trigger model measurement method. The trigger model is waiting on an event.
INACT	The trigger model is inactive. This occurs when the trigger model cannot run, such as when the count is more than the reading buffer capacity or if the buffer is style writable.

System event indicator

Located on the right side of the instrument status indicator bar, this indicator changes based on the type of event that has been logged.

Select the indicator to open a message screen with a brief description of the error, warning, or event. Select the Event Log button to open the System Events tab of the event log, which you can use to access detailed descriptions of the events.

For more information about the Event Log, see .

Figure 11: Error and message indicator
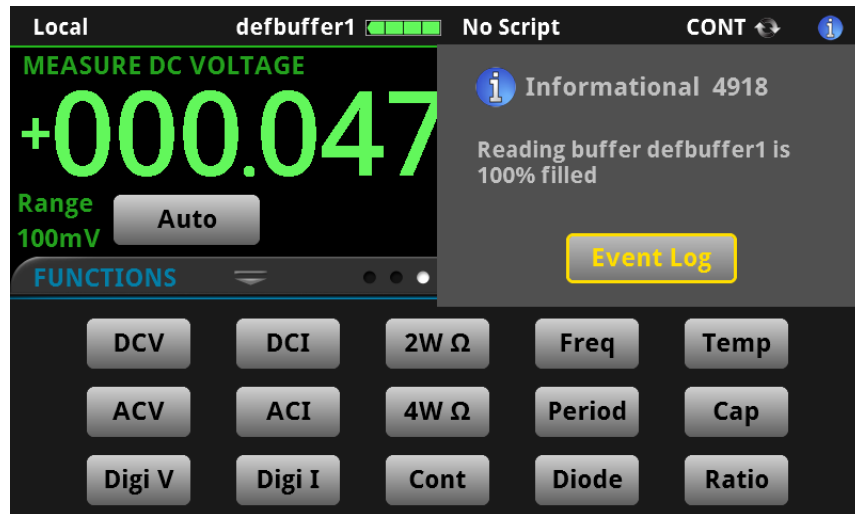
The following table describes the icons you may see here.

Icon	Description
	An empty triangle means that the no new events were logged in the event log since the last time you viewed the event log.
	A blue circle means that an informational event message was logged. The message is for information only. This indicates status changes or information that may be helpful. If the Log Command option is on, it also includes commands.
	A yellow triangle means that a warning event message was logged. This message indicates that a change occurred that could affect operation.
	A red triangle means that an error event message was logged. This may indicate that a command was sent incorrectly.

Measure view area

The Measure view area of the Home screen displays the value of the present measurement and other measurement information.

Figure 12: Measure view area of the Home screen

The Range button on the lower left displays the presently selected measure range. Select the button to change the range.

The indicators on the right edge of the Measure view area show any measure settings that affect the displayed measurement value. The indicators and what they mean are defined in the following table.

Indicator	Meaning
10MΩ	Input impedance is set to 10 MΩ
ACCPL	AC signal coupling is enabled
AUTOΩ	Input impedance is set to automatic
AZERO	Instrument automatically retrieves reference values
DCCPL	DC signal coupling is enabled
DRYCR	Dry circuit ohms is enabled
FILT	A filter is applied to the measurement
L1FAIL	Limit test one is enabled and measurement failed
L1PASS	Limit test one is enabled and measurement passed
L2FAIL	Limit test two is enabled and measurement failed
L2PASS	Limit test two is enabled and measurement passed
MATH	A percent, mx+b, or reciprocal calculation is applied
OCMP	Offset compensation is enabled
OLEAD	Open lead detection is enabled
REL	Relative offset is applied
SIMJC	The thermocouple reference junction is simulated

Interactive swipe screens

The Model DMM7510 touchscreen display has multiple screens that you can access by gently swiping left or right on the lower half of the display. The options available in the swipe screen are described in the following topics.

Swipe screen heading bar

The heading bar of the swipe screen contains the following options.

Figure 13: Model DMM7510 swipe screens, maximized and minimized
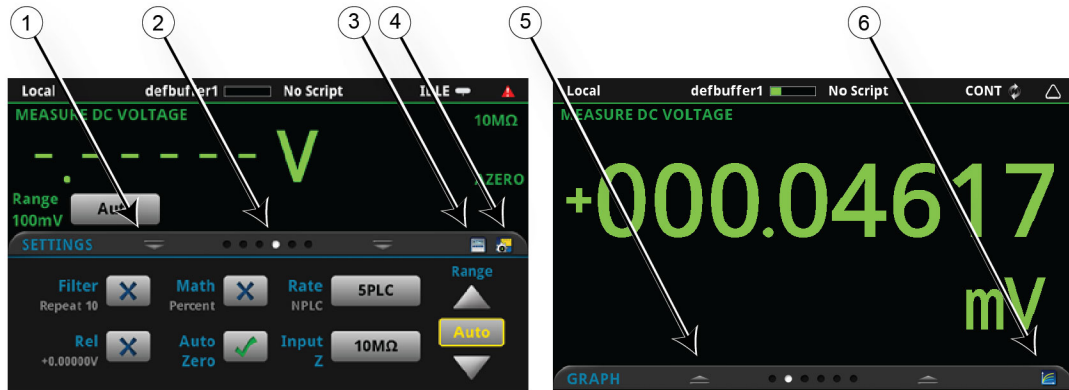
#	Screen element	Description
1	Minimize indicator	You can swipe down to minimize the swipe screens.
2	Swipe screen indicator	Each circle represents one swipe screen. As you swipe right or left, a different circle changes color, indicating where you are in the screen sequence. Select a circle to go to a swipe screen without swiping.
3	Calculations shortcut	Select to open the CALCULATIONS SETTINGS menu.
4	Settings shortcut	Select to open the MEASURE SETTINGS menu for the selected function.
5	Restore indicator	Indicates that you can swipe up to display the swipe screen.
6	Graph shortcut	Select to open the Graph screen.

FUNCTIONS swipe screen

The FUNCTIONS swipe screen highlights the selected measure function and allows you to select a different function.

Figure 14: FUNCTIONS swipe screen

SETTINGS swipe screen

The SETTINGS swipe screen gives you front-panel access to some instrument settings. It shows you the present settings and allows you to change, enable, or disable them quickly. The available settings depend on which measure function is active.

Figure 15: SETTINGS swipe screen
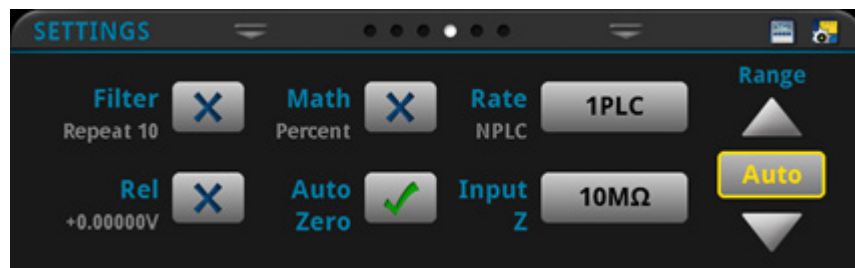
To disable or enable a setting, select the box next to the setting so that it shows an X (disabled) or a check mark (enabled).

For descriptions of the settings, use the navigation knob to select the button, then press the **HELP** key.

STATISTICS swipe screen

The STATISTICS swipe screen contains information about the readings in the active reading buffer. You can use the **Clear Active Buffer** button on this screen to clear the data from the active reading buffer.

Figure 16: STATISTICS swipe screen
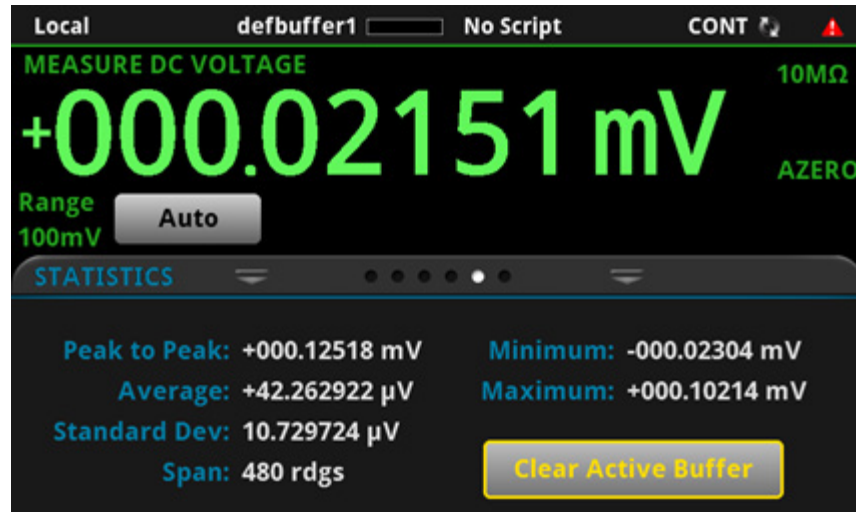
SECONDARY swipe screen

The SECONDARY swipe screen allows you to display the results of two measurements on the Home screen. Refer to

Figure 17: SECONDARY swipe screen
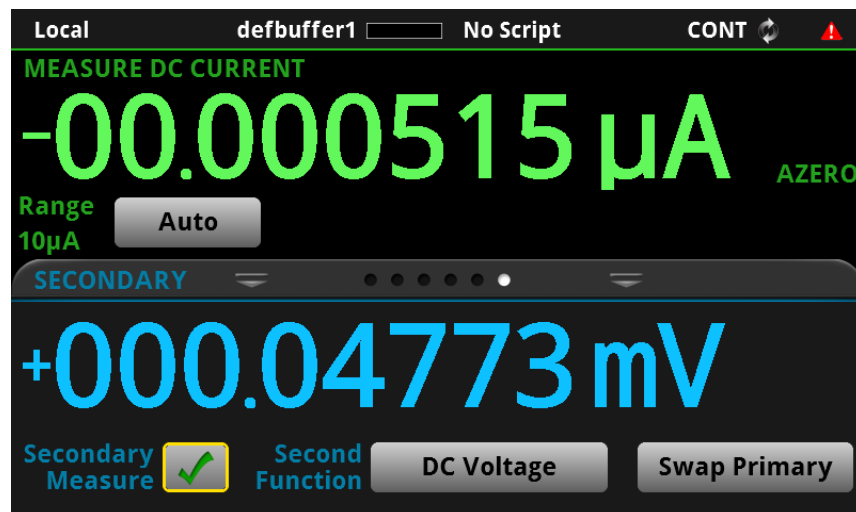
The secondary measurement window has the following control options:

Button	Description
Secondary Measure	Enables or disables the secondary measurement feature. The primary measurement is not affected by the state of the secondary measurement.
Second Function	Displays the list of functions so you can select the measure function for the secondary measurement.
Swap Primary	Switches the primary and secondary functions.

NOTE

Depending on the selected functions, a relay may click when the instrument switches between the measurement types. Leaving secondary measurements on for extended periods may shorten the life of the relays.

USER swipe screen

You can program custom text that appears on the USER swipe screen. For example, you can program the Model DMM7510 to show that a test is in process. For details about using remote commands to program the display, refer to

Figure 18: USER swipe screen
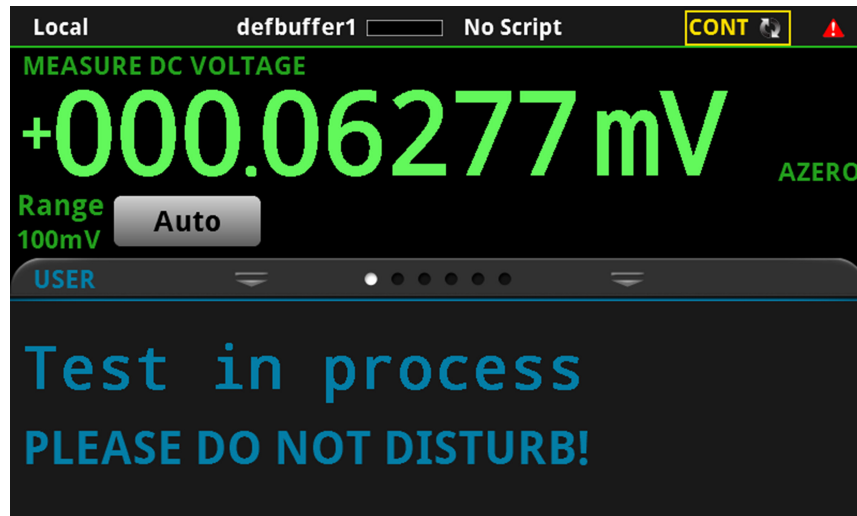
GRAPH swipe screen

The GRAPH swipe screen shows a graphical representation of the readings in the presently selected reading buffer.

Figure 19: GRAPH swipe screen
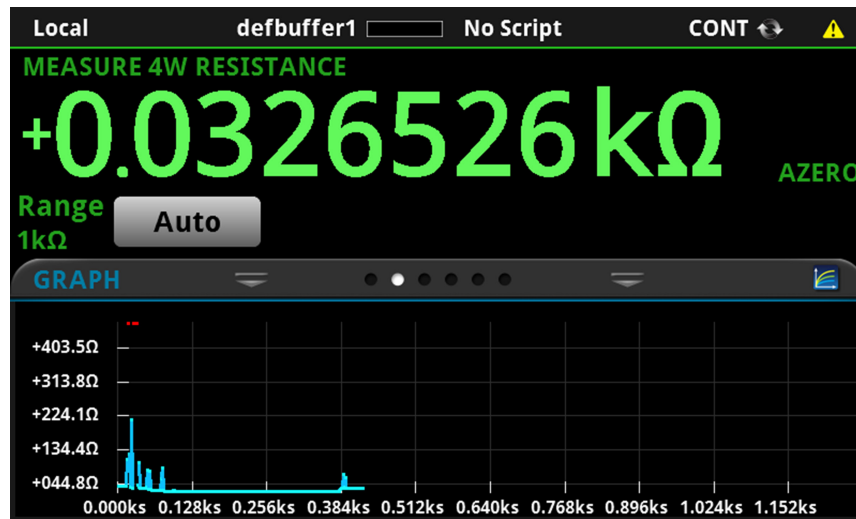
To view the graph in the full screen and to access graph settings, select the graph icon on the right side of the swipe screen header. You can also open the full-function Graph screen by pressing the **MENU** key and selecting **Graph** under Views.

Menu overview

To access the main menu, press the **MENU** key on the Model DMM7510 front panel. The figure below shows the organization of the main menu.

Figure 20: Model DMM7510 main menu
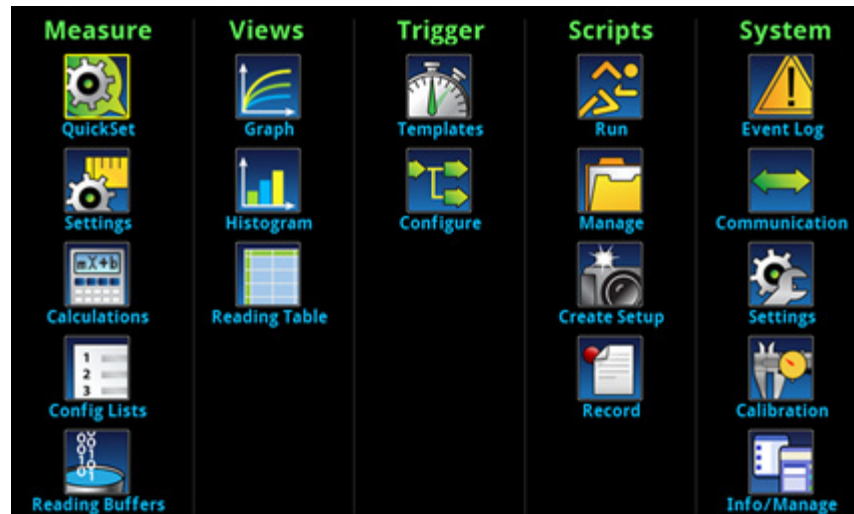
The main menu includes submenus that are labeled in green across the top of the display. Touching an icon in a submenu opens an interactive screen.

Measure menu

The Measure menus allow you to select, configure, and perform measure operations from the front panel. The following topics describe the settings that are available on these interactive screens.

QuickSet menu

The QuickSet icon at the top left of the main menu allows you to change the function, adjust performance, and set quick-setup options. You can also access the QuickSet menu by pressing the **QUICKSET** key on the front panel.

Setting	Description
Function	Selects the measure function that the instrument uses. Refer to DMM measurement overview (on page 2-91).
Performance	Adjusts the balance between resolution and speed of the instrument. Refer to Using the Performance slider (on page 2-132).
Quick Setups	Sets up specific measure tests by following a few prompts. Refer to Using Quick Setups (on page 2-132).

Measure Settings menu

The Measure **Settings** menu contains settings for the presently selected measure function, which is identified by the Function indicator in the upper right corner of the menu. The available settings depend on the front-panel **FUNCTION** key selection.

Function indicators

The Function indicator in the upper right corner of some menu screens displays which function the instrument is using to make measurements. The indicators include **DCV Ratio** to indicate that the DC voltage ratio function is selected and **2W Res** to indicate that the 2-wire resistance function is selected. You can select the indicator to open the list of functions and change the active function.

DC voltage measure settings

The following options are available on the Measure Settings menu when the function is set to DC voltage.

Setting	Description
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Input Impedance	Sets impedance to Auto or 10 MΩ. Refer to DC voltage input impedance (on page 2-95).
Unit	Allows voltage to be measured in volts (default) or decibels (to compress a large range of measurements into a smaller scope).
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).
Decibel Reference	Sets decibel reference point; this setting is only available when Unit is set to Decibel. See Show voltage readings in decibels (on page 2-95).

AC voltage measure settings

The following options are available on the Measure Settings menu when the function is set to AC voltage.

Setting	Description
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Detector Bandwidth	Sets the detector bandwidth. Refer to Detector bandwidth (on page 2-137).
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Line synchronization is always off for AC voltage measurements.
Unit	Allows voltage to be measured in volts (default) or decibels (to compress a large range of measurements into a smaller scope).
Decibel Reference	Sets decibel reference point; this setting is only available when Unit is set to Decibel. See Show voltage readings in decibels (on page 2-95).

DC current measure settings

The following options are available on the Measure Settings menu when the function is set to DC current.

Setting	Description
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).

AC current measure settings

The following options are available on the Measure Settings menu when the function is set to AC current.

Setting	Description
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Detector Bandwidth	Sets the detector bandwidth. Refer to Detector bandwidth (on page 2-137).
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2). For detector bandwidths of 3 Hz or 30 Hz, the setting is fixed to 16.67 ms (1 PLC).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147). For detector bandwidths of 3 Hz or 30 Hz, this setting is always on.
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Line cycle synchronization is always off for AC current measurements.

2-wire resistance measure settings

The following options are available on the Measure Settings menu when the function is set to 2-wire resistance.

Setting	Description/reference
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Offset Compensation	Always set to off for 2-wire resistance. Refer to Offset-compensated ohms (on page 2-108).
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).

4-wire resistance measure settings

The following options are available on the Measure Settings menu when the function is set to 4-wire resistance.

Setting	Description/reference
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Offset Compensation	Enables or disables offset compensation. When enabled, offset compensation reduces or eliminates thermoelectric EMFs in low-level resistance measurements. Refer to Offset-compensated ohms (on page 2-108).
Open Lead Detect	Enables or disables open lead detection. When enabled, detects open test leads, which can lead to inaccuracies in 4-wire sensing.
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).
Dry Circuit	Enables or disables dry circuit. When dry circuit is enabled, it limits the open-circuit voltage to below 20 mV. When dry circuit is enabled, offset compensation is automatically enabled. Refer to Dry circuit ohms (on page 2-108).

Continuity measure settings

The following options are available on the Measure Settings menu when the function is set to Continuity.

Setting	Description
Range	Always set to 1 kΩ when Continuity is selected.
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Always set to 4.5 digits when Continuity is selected.
Limit 1 High Value	Sets the high value for limit 1. Limit 1 is automatically set to enable. Refer to Limit testing and binning (on page 3-99).
Limit 1 Audible	Determines if the beeper sounds when the resistance is more than or less than the limit 1 high value. Refer to Limit testing and binning (on page 3-99).
Integration Rate	Always set to 0.006 PLC when Continuity is selected.
Auto Zero	Always set to off when Continuity is selected.
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).

Frequency measure settings

The following options are available on the Measure Settings menu when the function is set to Frequency.

Setting	Description
Range	Always set to auto when frequency is selected.
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Always set to 6.5 digits for frequency.
Aperture	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Count	Sets the number of readings to make when a measurement is requested.
Threshold Range	Indicates the expected input level of the voltage signal. You can also set the threshold range to Auto.
Threshold Level	Determines the signal level where the instrument makes frequency measurements.

Period measure settings

The following options are available on the Measure Settings menu when the function is set to Period.

Setting	Description
Range	Always set to auto when period is selected.
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Always set to 6.5 digits for period.
Aperture	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Count	Sets the number of readings to make when a measurement is requested.
Threshold Range	Indicates the expected input level of the voltage signal. You can also set the threshold range to Auto.
Threshold Level	Determines the signal level where the instrument makes period measurements.

Diode measure settings

The following options are available on the Measure Settings menu when the function is set to Diode.

Setting	Description
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4). Always set to 10 V for diode test.
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Bias Level	Sets amount of current that is sourced by the instrument to make measurements.
Limit 1 Low Value	Sets the low value for limit 1. Refer to Limit testing and binning (on page 3-99).
Limit 1 Audible	Determines if the beeper sounds when the measurement is more than or less than the limit 1 high value. Refer to Limit testing and binning (on page 3-99).
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1). Always set to off for diode test.
Limit 1 High Value	Sets the high value for limit 1. Refer to Limit testing and binning (on page 3-99).

Temperature measure settings

The following options are available on the Measure Settings menu when the function is set to Temperature.

Setting	Description
Unit	Set the type of units that are displayed on the front panel and stored with the temperature measurement in the reading buffer.
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Transducer	Sets the type of transducer that is used to temperature measurements.
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).
Thermocouple	Thermocouple only. Sets the thermocouple type.
Temperature	Thermocouple only: The simulated reference temperature.
Open Lead Detector	Thermocouple and RTDs only: Enables or disables open lead detection. When enabled, detects open test leads.
Reference Junction	Thermocouple only: Displays Simulated.
Thermistor	Thermistor only: Sets the type of thermistor.
RTD Type	RTD only: Sets the RTD type. See RTDs (Resistance Temperature Detectors).
Offset Compensation	RTD only: Enables or disables offset compensation.
RTD Alpha	RTD only when USER is selected: Sets the alpha value of a user-defined RTD.
RTD Beta	RTD only when USER is selected: Sets the beta value of a user-defined RTD.
RTD Delta	RTD only when USER is selected: Sets the delta value of a user-defined RTD.
RTD Zero	RTD only when USER is selected: Sets the zero value of a user-defined RTD.

Capacitance measure settings

The following options are available on the Measure Settings menu when the function is set to capacitance.

Setting	Description
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52). Fixed at 4.5 digits for capacitance.
Count	Sets the number of readings to make when a measurement is requested.

DC voltage ratio measure settings

The following options are available on the Measure Settings menu when the function is set to DCV Ratio.

Setting	Description
Range	Determines the full-scale input for the measurement in the numerator of the ratio. The range also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Auto Delay	Applies a wait period at the start of measurement to allow cables and circuitry to settle for best accuracy. Refer to Auto Delay (on page 2-134).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Sense Range	Determines the full-scale input for the reference measurement in the denominator of the ratio. It also affects the accuracy of the measurements and the maximum signal that can be measured. Autorange is automatically set to off if a specific value is set.
Integration Rate	Controls the amount of time the input signal is measured (aperture), which affects the noise and reading rate; see Using aperture or NPLCs to adjust speed and accuracy (on page 4-2).
Auto Zero	Determines if internal reference points are used to maintain stability and accuracy. See Automatic reference measurements (on page 2-147).
Count	Sets the number of readings to make when a measurement is requested.
Line Sync	Enables or disables line synchronization. When it is enabled, it helps increase common-mode and normal-mode noise rejection. Refer to Line cycle synchronization (on page 4-1).

Digitize voltage measure settings

The following options are available on the Measure Settings menu when the function is set to digitize voltage.

Setting	Description/reference
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Input Impedance	Sets impedance to Auto or 10 MΩ; see DC voltage input impedance (on page 2-95)
Signal Coupling	Determines if AC or DC signal coupling is used. When DC is selected, the instrument measures AC and DC components of the signal. When AC is selected, the instrument only measures the AC components of the signal.
Aperture	Sets the measurement aperture time.
Sample Rate	Sets the number of readings per second.
Count	Sets the number of readings to make when a measurement is requested.
Unit	Sets the units of measurement that are displayed on the front panel of the instrument and stored in the reading buffer.
AC Coupling Filter	Available when Signal Coupling is set to AC: Sets the instrument settling time. Set to Slow to allow more settling time or Fast to allow less settling time.
AC Coupling Frequency	Available when Signal Coupling is set to AC: Sets an amplitude that compensates for signal loss across the coupling capacitor.
Decibel Reference	Sets decibel reference point; this setting is only available when Unit is set to Decibel. See Show voltage readings in decibels (on page 2-95).

Digitize current measure settings

The following options are available on the Measure Settings menu when the function is set to digitize current.

Setting	Description/reference
Range	Determines the full-scale input for the measurement; also affects the accuracy of the measurements and the maximum signal that can be measured. Refer to Ranges (on page 3-4).
Display Digits	Sets the number of digits that are displayed for front-panel readings. It does not affect accuracy or speed. Refer to Setting the number of displayed digits (on page 2-52).
Aperture	Sets the measurement aperture time.
Sample Rate	Sets the number of readings per second.
Count	Sets the number of readings to make when a measurement is requested.

Measure Calculations menu

The **Calculations** menu contains settings that specify the way measurement information is processed and returned.

Setting	Description
Rel	Use the relative offset feature to subtract a set value or a baseline reading from measurement readings. When you enable relative offset, all subsequent measurements are displayed as the difference between the actual measured value and the relative offset value
Filter	Enables or disables the averaging filter for measurements of the selected function.
Config	Displays the settings that are available for the averaging filter.
Filter Type	Selects the type of averaging filter that is used for the selected measure function when the measurement filter is enabled. Select the moving average filter to continuously add measurements to the stack on a first-in, first-out basis, replacing the oldest measurement in the stack with a new measurement. Select the repeating average filter to average a set of measurements and then flush the data out of the stack before averaging a new set of measurements.
Filter Count	This sets the number of measurements that are averaged when filtering is enabled.
Filter Window	This sets the window for the averaging filter that is used for measurements for the selected function.
Limits	These options allow you to do pass-or-fail limit testing through the front panel. When you do limit testing and a test fails, the limit (1 or 2) that failed is shown on the Home screen.
Math	This setting enables or disables math operations. When this is on, the math operation specified by Math Function is applied to the measurement.
Config	Displays the settings that are available for the math functions.
Math Format	When Math is enabled, you can specify which math operation is performed on measurements. You can choose one of the following math operations: • mx+b: Manipulate normal display readings by adjusting the m and b factors. • Percent: Specify a constant that is applied to the measurement and display readings as percentages. • Reciprocal: The reciprocal math operation displays measurement values as reciprocals. The displayed value is $1/X$, where X is the measurement value (if relative offset is being used, this is the measured value with relative offset applied).
m(Scalar)	Defines the constant for the scale factor.
b(Offset)	Defines the constant for the offset factor.
Zero Reference	When the Math State is set to On, this setting specifies the reference used when the math operation is set to percent; the range is -1e12 to +1e12.

Limit 1 and Limit 2	These settings enable or disable limit testing. The Limit options allow you to do pass-or-fail limit testing using the front panel of the instrument. When you do a limit test, the Home screen displays the pass or fail result of the test.
	Limit State enables or disables a limit test for the selected measurement function. When testing is enabled, limit testing occurs on each measurement. Limit testing compares the measurements to the high and low limit values. If a measurement is outside these limits, the test fails. If a measurement is in the limits, it passes.
Config	Displays the settings that are available for the limit functions.
Auto Clear	Auto clear indicates if the high and low limits should be cleared automatically or not.
Low Value	The Low Value specifies the lower limit for limit tests.
High Value	The Limit High Level specifies the upper limit for a limit test.
Audible	Audible determines if the instrument beeper sounds when a limit test passes or fails, or disables the beeper.

Measure Config Lists menu

The **Config Lists** menu allows you to select an existing measure configuration list, create a new list, load configuration settings to and from the instrument (system), and view the settings of an index in a configuration list. For more information about using configuration lists, see Configuration lists (on page 3-32).

Setting	Description
Select List	Displays a list of available configuration lists from which you can choose.
New List	Creates a new, empty configuration list. To populate the list with the present instrument settings, select **System to Index**.
Delete Index	Deletes a configuration list index from the selected configuration list.
System to Index	Saves the present instrument settings to an index at the end of the selected configuration list.
Index to System	Restores the instrument to the settings stored in the selected configuration list index.
Index Details	Displays the details of the selected configuration index. Details include settings such as function, value, delay, calculations, and range. You can scroll the displayed list to view additional settings.

Measure Reading Buffers menu

The **Reading Buffers** menu allows you to view the list of existing reading buffers and select one to be the active buffer. You can also create, save, delete, resize, and clear buffers from this screen.

To create a new reading buffer, select New. The new buffer is automatically set to be the active buffer.

To adjust settings for a specific buffer, select the buffer. The Settings screen for that buffer is displayed. A brief description of the options is provided in the following table.

Setting	Description
Capacity	Sets the maximum number of readings that the buffer can store. Note that when you resize a buffer, the readings contained in that buffer are cleared.
Fill Mode	**Continuous:** Fills the buffer continuously and overwrites old data when the buffer is full. **Once:** Stops collecting data when the buffer is full (no data is overwritten).
Clear	Clears data from the selected buffer.
Make Active	Makes the selected buffer the active reading buffer.
Save to USB	Saves the buffer to a `.csv` file, which can be opened by a spreadsheet program. A USB flash drive must be present in the front-panel USB port before you select Save to USB.
Delete	Deletes the selected buffer.

Views menu

The menus under Views in the main menu allow you to select, configure, and view data from measure operations on the Model DMM7510. The following topics describe the settings that are available on these interactive screens.

Views Graph menu

The **Graph** menu opens a screen that displays a graph of the measurements in selected reading buffers as traces. It also contains tabs that you use to customize the graph display.

You can also select the trigger mode and initiate the trigger model from this screen. Select the Trigger mode indicator in the upper right corner of the screen and select the trigger mode. Refer to Trigger mode indicator (on page 2-15).

Graph tab

The Graph tab graphs readings as they are made by the instrument. Settings you make on the Data, Scale, and Trigger tabs affect how readings appear on this screen. You can also select the number of traces that are displayed.

Figure 21: Model DMM7510 Graph tab
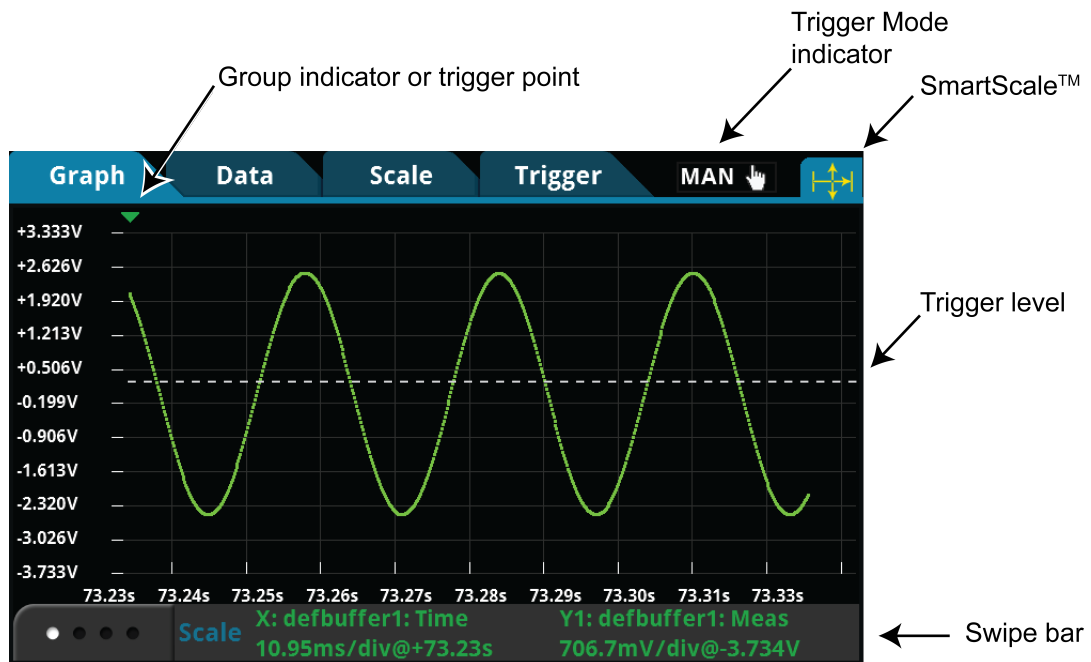
You can zoom in or out in the graph view by placing two fingers on the screen and moving them together or apart in a pinching motion. You can also move the view of the graph to the left or right by placing a finger on the screen and moving it in either direction. If you want to set the method of scaling data to SmartScale™, select the icon in the upper-right corner of the Graph tab. The instrument determines the best way to scale data based on the data and the instrument configuration (such as the measure count).

You can set the X and Y axes to show different values appropriate for your application. The bottom of the Graph tab contains a legend of the active axis and scale settings for the graph.

Data tab

The Data tab allows you to select the reading buffer that provides the data that is displayed on the Graph tab. You can select up to four buffers. The data from each buffer is shown as a separate trace on the Graph tab. You can also select the type of marker that is used on the graph.

Setting	Description
Trace Data	Displays the names of the reading buffers that are displayed on the Graph tab. If no trace is selected, the active buffer is used. You can select up to four reading buffers. The data from each buffer is displayed as a separate trace on the graph. If you select the active buffer, the trace is set to be the reading buffer and the label "Active" is removed. The Graph tab will no longer switch to the new buffer if you change the active buffer. To return to plotting data from the active buffer, use Remove Trace to remove all of the traces. When the active buffer setting is on, "Active" is displayed before the name of the active buffer. When a trace is using the active buffer and the active buffer changes, the graph replots the data from the newly designated active buffer.
Add Trace	Selects a reading buffer that is used to supply the data for a trace Graph tab. You can select up to four reading buffers. The colors for the traces are initially established in the order green, blue, orange, and brown. Once a color has been assigned to a trace, the color remains associated with that trace. For example, trace 3 is assigned to orange. If trace 1 and 2 are removed, trace 3 becomes trace 1, but remains orange on the Graph tab. New traces are assigned the next available color. In this example, if you add a trace, it is set to green.
Remove Trace	Removes the trace that is selected in the Trace Data list. This removes the trace from display on the Graph tab. If the active buffer is turned off ("Active" is not displayed before one of the buffer names), removing all traces turns the active buffer setting on.
Clear Buffer	Clears data from the selected buffer.
X-Axis	Displays the reading buffer and units that are used for the X-axis.
Marker	The markers determine how data is represented when there are many data points. You can select Min/Max, Average, or Both. When Min/Max is selected, the graph displays the minimum and maximum points of the data as a solid line. When Average is selected, the data is averaged and displayed as a single averaged data point. When Both is selected, the Min/Max line and the Average data point are displayed.

Scale tab

The Scale tab contains settings that allow you to fine-tune the output on the Graph tab.

Setting	Description
X-Axis Method	The method determines how data is scaled and tracked on the Graph tab. You can select: • SmartScale: The instrument determines the best way to scale data based on the data and the instrument configuration (such as the measure count). • Track Latest: The graph always shows the latest data. • Track Group: The graph always shows the entire data group on the graph. • All: All data is displayed on the graph. • Off: The graph is not automatically adjusted.
X-Axis and Y-Axis Scale	Sets the reading value scale for each division.
X-Axis and Y-Axis Minimum Position	Sets the first value that is visible on the graph for the selected trace.
Trace	Toggles between the available traces. Information specific to the trace is shown in the same color as the trace.
Y-Axis Method	The scale method determines how data is scaled on the Graph tab. If you are graphing one trace, you can select: • SmartScale: The instrument scales the graph automatically. The scale is set to fit all the data that is in the selected reading buffer onto the screen. The instrument determines the best scale based on the data. • Off: No automatic resizing occurs. You can adjust the data manually by swiping, pinching, and zooming. If you are graphing multiple traces, you can select: • SmartScale: The instrument scales the graph automatically. The instrument determines the best scale and tracking method based on the data, reading groups, number of traces, and instrument configuration. • Per Trace: Each trace is displayed on the graph. • Lanes: Each trace is displayed in a separate partition of the graph. When the Lanes option is selected, there are no measurement scale axes, only time per division. • Shared: Accommodates the minimum and the maximum of all traces. • Off: No automatic scaling.
Y-Axis Scale Format	Sets the scale format that is used on the graph. Select Linear to increase the step size in even increments. Select Log to increase the step size exponentially.

Trigger tab

The Views Graph Trigger tab contains settings that define the trigger mode.

The Trigger Mode button allows you to select a predefined trigger model. Refer to Trigger mode indicator (on page 2-15).

Setting	Description
Source Event	Determines the event that is used to trigger measurements. You can select: • Display TRIGGER Key: The trigger occurs when you press the TRIGGER key. • External: The trigger occurs when an external pulse is detected. The external pulse can come from a digital input line, TSP-Link input line, or the rear-panel external input line. • Waveform: Select an analog edge, pulse, or window to trigger.
Delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay.
Position	The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled.
Edge	When the source is set to Digital, TSP-Link, or External, this sets the type of edge that generates a trigger. You can set it to rising, falling, or either.
Digital In Line	When the source is set to Digital, this selects the digital input line that will generate the trigger (1 to 6).
TSP-Link Line	When the source is set to TSP-Link, this selects the TSP-Link input line that will generate the trigger (1 to 3).
Level	When the analog edge waveform is selected, sets the signal level that generates the trigger event. The level can be set to any value within the selected measurement range.
Slope	When the analog edge waveform is selected, sets the slope to rising or falling. Rising causes a trigger event when the analog signal trends from below the analog signal level to above the level. Falling causes a trigger event when the signal trends from above to below the level.
High Frequency Rejection	When the analog edge or pulse waveform is selected, enables or disables high frequency rejection.
Level	When the analog pulse waveform is selected, this defines the pulse level that generates a trigger event.
Condition	When the analog pulse waveform is selected, the condition defines if the pulse must be greater than or less than the pulse width before a trigger is generated.
Polarity	When the analog pulse waveform is selected, this determines if the trigger occurs when the pulse is above the defined signal level or below the defined signal level.
Pulse Width	When the analog pulse waveform is selected, this sets the pulse width in seconds. This option sets either the minimum or maximum pulse width that generates a trigger event. The value of pulse condition determines whether this value is interpreted as the minimum or maximum pulse width.
Low Boundary	When the analog window waveform is selected, this sets the lower boundary of the analog trigger window.
High Boundary	When the analog window waveform is selected, this sets the high boundary of the analog trigger window.
Direction	When the analog window waveform is selected, this defines if the analog trigger occurs when the signal enters or leaves the defined upper and lower analog signal level boundaries.

Views Histogram menu

The **Histogram** menu allows you to graph the distribution of measurement data in the selected reading buffer. It also contains tabs that you use to customize the histogram.

Histogram tab

The Histogram tab graphs readings as a bar graph of the data distribution into bins. Settings you make on the Data and Scale tabs affect which data are used and how data distributions appear on this screen. You can change the scale of either axis on the screen by dragging or pinching the screen.

Figure 22: Model DMM7510 Histogram
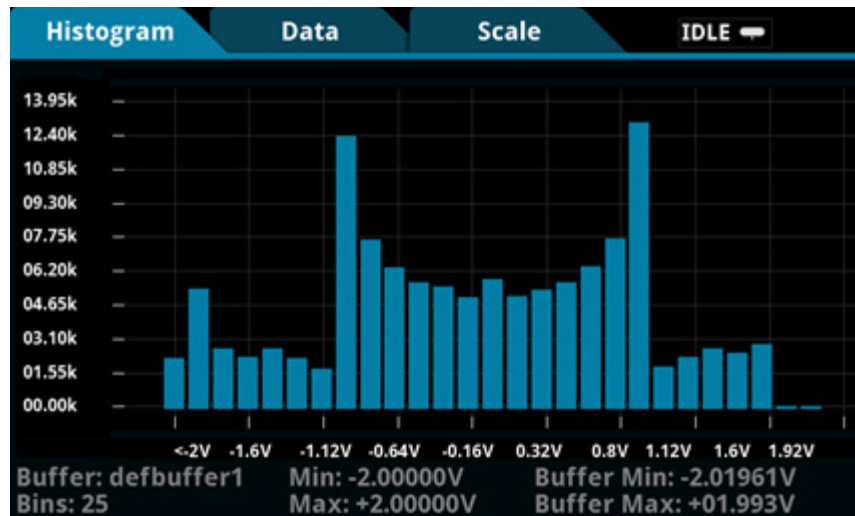
Data tab

The Data tab allows you to select which reading buffer provides the data that is binned on the Histogram tab. You can also clear the data from the selected buffer.

Setting	Description
Bin Buffer	The presently selected reading buffer. You can use this button to select a different reading buffer.
Clear Buffer	Clear the reading buffer. The Histogram is also cleared.

Scale tab

The Scale tab allows you to set up boundaries, number of bins, and type of scaling used for the histogram.

Setting	Description
Minimum Boundary	The lowest value of the data that is binned in the histogram. Data that is below this level is binned in the low outlier bin.
Maximum Boundary	The highest value of the data that is binned in the histogram. Data that is above this level is binned in the high outlier bin.
Number of Bins	The number of bins in the histogram. The histogram will create two outlier bins in addition to the bins you define. These bins are used to collect data that is below or above the defined minimum and maximum boundaries.
Auto Scale	Set the histogram to scale automatically or turn automatic scaling off.

Views Reading Table menu

The **Reading Table** menu allows you to view data in the selected reading buffer.

Setting	Description
Buffer	Selects the reading buffer that contains the data you want to view. If Active Buffer is selected, the data from reading buffer that is presently storing readings is displayed.
Up and down arrows	Selects the up or down arrow to jump up or down one page in the data displayed on the screen.
Jump to	Jumps to the row you specify in the reading table. This is useful to avoid scrolling through large lists of values.
Refresh	Refreshes the list with the most recent data for the readings in the selected reading buffer.
Table	Displays the data in the selected reading buffer. You can select a data point to display additional detail about that data point.
Reading Details	Select a data point to opens the Reading Details window for the selected data point. The details describe the instrument settings when the data point was read.

Trigger menu

The menus under Trigger in the main menu allow you to configure triggering operations from the Model DMM7510 front panel. The following topics describe the settings that are available on these interactive screens.

Trigger Templates menu

The **Templates** menu allows you to choose from one of several preprogrammed trigger models. When you select a template, settings you can specify for that template are shown in the lower part of the screen.

You can also customize the templates from the front panel using the Configure menu under Trigger on the main menu screen. For details, see .

The table below describes the trigger model templates and available user-specified settings.

Setting	Description
Empty	Clears the present trigger model.
ConfigList	Creates a trigger model that loads a configuration list. At each configuration list point, a measurement is made. The list is iterated until every point in the configuration list has been loaded.
ExtDigTrigger	Creates a trigger model that waits on an input line, delays, makes a measurement, and sends out a trigger on the output line a specified number of times.
SimpleLoop	Creates a trigger model that sets up a loop that sets a delay, makes a measurement, and then repeats the loop the number of times you define in the count parameter.
DurationLoop	Creates a trigger model that makes continuous measurements for a specified amount of time.
LoopUntilEvent	Creates a trigger model that makes continuous measurements until a specified event occurs.
GradeBinning	Creates a trigger model that successively measures components and compares their readings to high or low limits to grade components.
SortBinning	Creates a trigger model that successively measures components and compares their readings to high or low limits to sort components.
Keithley2001	Creates a multi-layer trigger model that emulates the Keithley Instrument Model 2001 trigger model.

Trigger Configure menu

The **Configure** menu allows you to see and modify the structure and parameters of a trigger model. You can also monitor trigger model operation.

Figure 23: Model DMM7510 TRIGGERFLOW screen
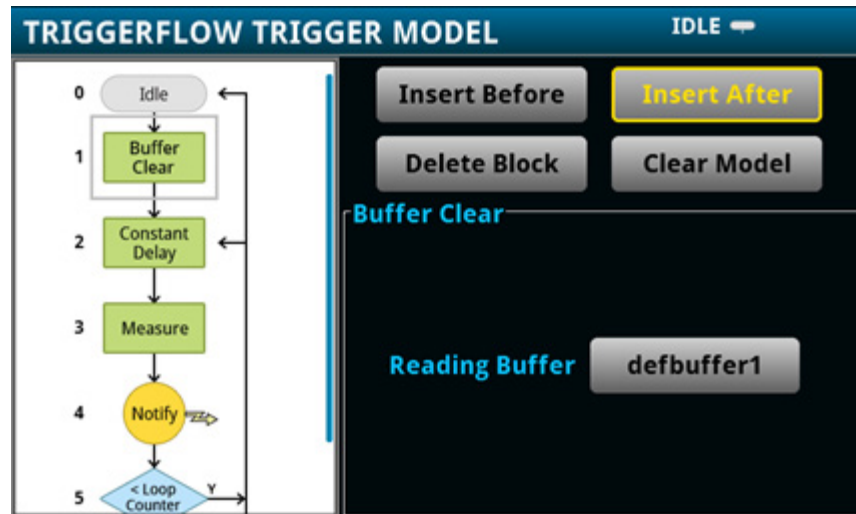
To see the parameters that you can change from the front panel, select a block in the trigger model diagram. The available options change depending on the type of block you select.

From this screen, you can:

- Insert a new trigger block before or after the selected block
- Choose among several block types to add
- Edit an existing block
- Delete an existing block
- Remove all trigger blocks by selecting Clear Model

When you finish your changes to the trigger model, you can initiate the trigger model by pressing the front-panel TRIGGER key.

Scripts menu

The menus under Scripts in the main menu allow you to configure, run, and manage scripting operations from the Model DMM7510 front panel. Scripts are blocks of commands that the instrument can run as a group. The following topics describe the settings that are available on these screens.

Scripts Run menu

The **Run** menu contains a list of scripts that you can select to run immediately. You can also copy a script to a script that runs each time the instrument power is turned on. You can access scripts that are in the instrument or on a USB flash drive.

Setting	Description
Available Scripts	Displays a list of available scripts that you can select. All scripts that are saved on the Model DMM7510 or are on a USB flash drive inserted into the instrument are listed.
Run Selected	Runs the selected script immediately.
Copy to Power Up	Saves the selected script to a script that runs automatically when the instrument is turned on. The script is saved with the script name `autoexec`.

Scripts Manage menu

The **Manage** menu allows you to copy scripts to and from the instrument and the USB flash drive. You can also delete scripts from the instrument or USB flash drive.

Setting	Description
>	Copies a script from the instrument to a USB script. A USB flash drive must be inserted before you select this option.
<	Copies a script from a USB flash drive to the instrument. A USB flash drive must be inserted before you select this option.
Delete	Deletes the script that is selected.

For more information about using scripts with the Model DMM7510, see Fundamentals of scripting for TSP (on page 7-4).

Figure 24: Model DMM7510 MANAGE SCRIPTS menu
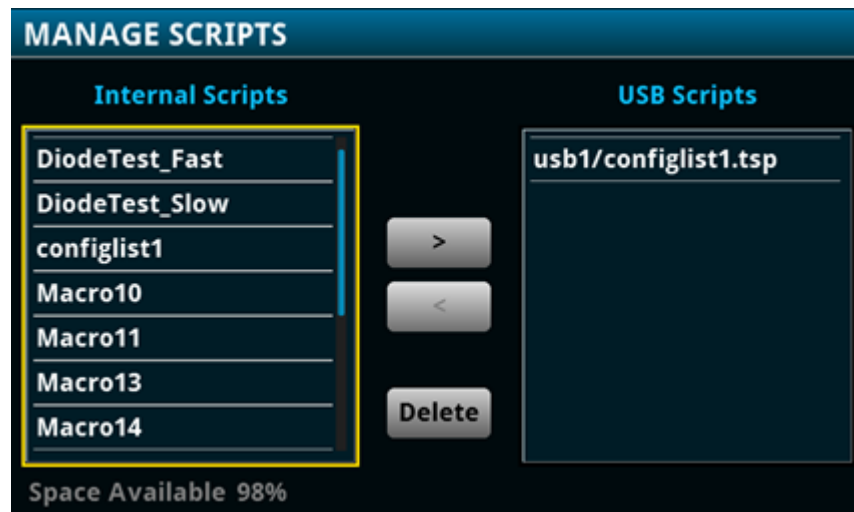
Scripts Create Setup menu

The **Create Setup** menu allows you to save the present settings and configuration lists of the instrument into a configuration script. You can use this script to recall the settings.

For more information about user configuration scripts and setups, see Saving setups (on page 2-148).

Figure 25: Model DMM7510 CREATE SETUP menu
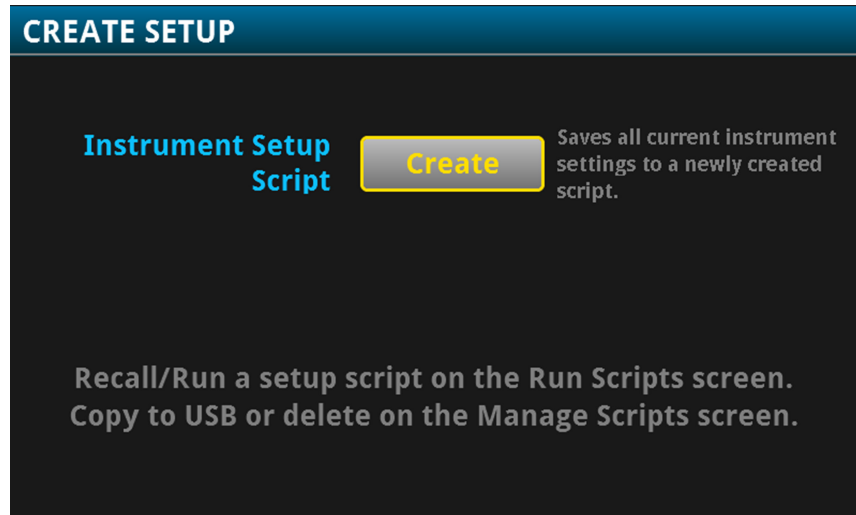
Scripts Record menu

The options in the **Record** menu allow you to record your actions and store them in a macro script. The script can be run and managed like any other script using the options in the Scripts menu or remote commands. Note that only settings are stored; no key presses or front-panel only options are stored.

Setting	Description
Start Macro	Begin recording your selections.
Stop Macro	Stop recording. You are prompted to enter a Macro Script Name.
Cancel Macro	Stop recording without saving.

System menu

The menus under System in the main menu allow you to configure general instrument settings from the Model DMM7510 front panel. Among these settings are the event log, communications, backlight, time, and password settings.

The following topics describe the settings that are available on these interactive screens.

System Event Log menu

The **Event Log** menu allows you to view and clear event log entries. You can also adjust which events are displayed or logged.

The System Events tab view shows event log entries in a table. Select a line in the table to open a dialog box that contains more detailed information about the event. The event log entries are one of the following types:

- **Error:** An error occurred. This may indicate that a command was sent incorrectly.
- **Warning:** This message indicates that a change occurred that could affect operation.
- **Information:** The message is for information only. This indicates status changes or information that may be helpful. If the Log Command option is on, it also includes commands.

The Log Settings tab view contains settings that affect what data displays on the System Events tab. The following table describes these settings.

Setting	Description
Show Warning	Turns the display of warnings on or off. If you turn this off, the instrument continues to record warnings and display warning popup messages, but does not display them on the System Events tab.
Show Information	Turns the display of information messages on or off. If you turn this off, the instrument continues to record information messages and display popup messages, but does not display them on the System Events tab.
Popups	Chooses what type and whether to display popup messages on the front panel. You can choose to display error messages, error and warning messages, or no messages in popups. Messages continue to be saved in the event log when popups are turned off.
Log Warning	Turns the logging of warnings on or off. If this is turned off, the instrument does not log or display popups for warning messages.
Log Information	Turns the logging of information messages on or off. If this is turned off, the instrument does not log or display popups for information messages.
Log Command	Turns the logging of commands on or off. When logging is turned on, the instrument records the commands that are sent to the instrument. It records commands sent from any interface (the front panel or a remote interface).
Reset Popups	Restores the popups setting to show errors and warnings.
Save to USB	Saves the event log to a .csv file on the USB flash drive. The filename is `eventlog.csv`.
Clear Log	Clears all entries from the event log.

System Communication menu

The **Communication** menu opens a set of tabs that contain information about the Model DMM7510 communications settings. Most of the tabs contain settings that you can change.

GPIB tab setting	Description
Address	The default GPIB address is 16. You can set the address to any address from 1 to 30 if it is unique in the system. This address cannot conflict with an address that is assigned to another instrument or to the GPIB controller.

USB tab	No settings available for USB.

LAN tab setting *	Description
TCP/IP Mode	Select Auto to set the instrument to automatically obtain an IP address. Select Manual to manually set the IP address, gateway, and subnet values.
IP Address	Displays the present IP address. When TCP/IP Mode is set to Manual, you can set the IP address. To change the address, select the button next to IP Address and enter a new address.
Gateway	Displays the present gateway address. When TCP/IP Mode is set to Manual, you can set the gateway address. To change the address, select the Gateway button and enter a new address.
Subnet	Displays the present subnet mask address. When TCP/IP Mode is set to Manual, you can set the subnet mask address. To change the address, select the Subnet button and enter a new address.
Apply Settings	To save any changes you made on the LAN tab, select **Apply Settings**.
MAC Address	Read-only text that shows the present MAC address of the instrument.
* To save LAN settings, you must select **Apply Settings** after making any changes.	

TSP-Link tab setting	Description
Node	Select the button next to Node to set the TSP-Link node number for the instrument (1 to 64). Each instrument or enclosure attached to the TSP-Link expansion interface is called a node. Each node must be identified with a unique node number. This identification is called a TSP-Link node number.
Initialize	Select **Initialize** to have the Model DMM7510 find all connected TSP-Link instruments and form a network.

System Settings menu

The **Settings** menu contains general instrument settings.

Setting	Description
Audible Errors	Turns the beeper on or off. When the beeper is on, the beeper sounds when an event or error occurs. The audible error setting is not affected by instrument reset or power cycle. For more information, see Instrument sounds (on page 2-88).
Key Click	Turns the sound that occurs when you press a front-panel key On or Off. The key-click setting is not affected by instrument reset or power cycle.
Backlight Dimmer	Sets the front-panel display to dim after a period of time (1, 4, or 8 hours) or to never dim.
Backlight Brightness	Adjusts the brightness of the front-panel display. The sliding adjustment scale adjusts the brightness level.
Time and Date	Sets the instrument month, day, year, and time.
Command Set	Select the type of commands to use when controlling the instrument from a remote interface (SCPI or TSP).
Password	Contains the password if the instrument is set to use an access mode that requires a password. The Model DMM7510 is programmed with a default user name and password (case-sensitive): • User name: `admin` • Password: `admin` You can change the password. See Instrument access (on page 3-1) for more information about controlling access to the instrument.
Reading Format	Sets the format of the front-panel readings to Prefix (adds a prefix to the units symbol, such as k, m, or μ) or Exponent.
Access Mode	You can specify that the control interfaces request access before taking control of the instrument. There are several modes of access: Full, Exclusive, Protected, and Lockout. For details, see Instrument access (on page 3-1).
Line Frequency	The line frequency detected by the instrument. The line frequency is automatically detected and cannot be changed.

System Calibration menu

The **Calibration** menu allows you to start or manage auto calibration. Auto calibration removes measurement errors that are caused by the effects of temperature and time on components. You can also review factory adjustment and verification dates.

Option	Description
Last Run	Displays the date and time when auto calibration was last run.
Count	The number of times that auto calibration has been run since the last factory calibration. The count restarts at 1 after a factory calibration.
Warmup	Displays "Ready" if the instrument warmup period is complete. If the warmup period is incomplete, displays "Not Ready" and the time remaining.
Temperature Difference	The difference between the temperature of the instrument in Celsius (°C) compared to the temperature during the last auto calibration. The instrument updates the temperature when autozero references are refreshed. If autozero is set to off, the internal temperature is not updated.
Start ACAL	Runs auto calibration.
Scheduling Action	The action to take when the Scheduling Interval occurs: • Run: Run auto calibration. • Notify: Display a message that auto calibration should be run. • None: No scheduled auto calibrations occur.
Scheduling Interval	How often the Scheduling Action occurs. Only available when the scheduling action is set to Run or Notify.
Scheduled Time	The time when the Scheduling Action occurs, entered in 24-hour time format in one-hour increments. Only available when the scheduling interval is set to a day or more.
Next Run	Displays the date and time when the next Scheduling Action will occur.
Adjust Date	The date when the instrument was adjusted through a factory calibration.
Adjust Count	The adjustment count is the number of times the instrument has been factory calibrated.
Calibration Date	The date when the instrument calibration was last verified.

System Info/Manage menu

The **Info/Manage** menu gives you access to version and serial number information and settings for instrument firmware and reset functions.

Setting	Description
Version	The version of firmware that is installed in the instrument.
Serial Number	The serial number of the instrument.
Upgrade to New	This option initiates a firmware upgrade from a file on a USB flash drive.
Downgrade to Older	This option returns the Model DMM7510 to a previous version of the firmware from a file on a USB flash drive.
System Reset	This option resets many of the instrument settings to their default values.
Password Reset	Resets the access password to the default value.
Product Demo	Configures a brief demonstration of the Model DMM7510. To get correct results, you must have the appropriate demonstration fixture connected to the inputs. For more information, contact your local Keithley office, sales partner, or distributor.

Examples in this manual

Many of the remote interface examples in this manual show only one function. The features may be available for additional functions.

For example, many allow you to change the display digits.

The SCPI example shows only the DC voltage display digits command:

```
:DISPlay:VOLTage:DIGits 4
```

The example to change the number of displayed digits for TSP is shown as:

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.displaydigits = dmm.DIGITS_4_5
```

You can replace the SCPI `VOLTage` parameter or TSP `dmm.FUNC_DC_VOLTAGE` parameter with the parameter for another function to set the display digits for that function.

The function parameters for SCPI are shown in the following table.

VOLTage[:DC]	TEMPerature	RESistance	VOLTage[:DC]:RATIO
VOLTage:AC	CONTinuity	FRESistance	DIGitize:VOLTage
CURRent[:DC]	FREQuency[:VOLTage]	DIODe	DIGitize:CURRent
CURRent:AC	PERiod[:VOLTage]	CAPacitance	

The function parameters for TSP are shown in the following table.

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Display features

You can set the front-panel display to display the units of measure, number of digits, and customized text messages for your applications.

Setting the number of displayed digits

You can change the number of digits that are displayed for measurement readings on the front panel for some of the functions. For temperature, frequency, period, continuity, and capacitance, the displayed digits are fixed and cannot be changed.

The number of displayed digits does not affect accuracy or speed. It also does not affect the format of readings that are returned from a remote command. To change the format of remote interface readings for Test Script Processor (TSP) commands, see format.asciiprecision (on page 8-211). The format of remote interface readings for SCPI commands cannot be changed.

From the front panel:

1. Press **MENU**.
2. Under Measure, select **Settings**.
3. Next to Display Digits, select the number of digits to display.

This setting takes effect the next time you make a measurement.

From a remote interface:

- SCPI commands: Refer to :DISPlay:<function>:DIGits (on page 6-39).
- TSP commands: For measure functions, refer to dmm.measure.displaydigits (on page 8-143). For digitize functions, refer to dmm.digitize.displaydigits (on page 8-84).

Setting the display format

You can set the format of units that are displayed for measurement readings on the front panel. The formats are:

- Prefix: Add a prefix to the units symbol, such as k, m, or µ
- Exponent: Replace the units symbol with exponents

See the following figures for examples of each display format.

Figure 26: Model DMM7510 prefix display format

Figure 27: Model DMM7510 exponent display format

Set the displayed digits using the front panel

From the front panel:

Press the **MENU** key.

1. Under System, select **Settings**.
2. Select the button next to Reading Format.
3. Select the reading format (**Prefix** or **Exponent**).

This setting takes effect the next time you make measurements.

Over a remote bus:

*   SCPI commands: Refer to :DISPlay:READing:FORMat (on page 6-42)
*   TSP commands: Refer to display.readingformat (on page 8-64)

Customizing a message for the USER swipe screen

You can customize the message that is displayed on the USER swipe screen.

You must use a remote interface to customize the USER swipe screen.

Creating a message

When you create the message, you can send text that will be used on the top and bottom lines of the USER swipe screen. The top line allows up to 20 characters and the bottom line allows up to 32 characters.

The examples shown here switch the display to the USER swipe screen, set the first line to read "Test in process", and the second line to display "Do not disturb".

Using SCPI commands:

Send the commands:

```
DISPlay:SCReen USER
DISPlay:USER1:TEXT "Test in process"
DISPlay:USER2:TEXT "Do not disturb"
```

Using TSP commands:

Send the commands:

```
display.changescreen(display.SCREEN_USER_SWIPE)
display.settext(display.TEXT1, "Test in process")
display.settext(display.TEXT2, "Do not disturb")
```

Clearing the USER swipe screen

You can clear the message that is displayed on the USER swipe screen.

Using SCPI commands:

Send the command:

```
:DISPlay:CLEar
```

Using TSP commands:

Send the command:

```
display.clear()
```

Creating message for interactive prompts

If you are using the TSP command language and scripts, you can set up scripts that can prompt the operator to enter information from the front-panel display of the instrument.

The options that you can define include:

- Display a number pad so that operator can enter a value.
- Display a custom button that the operator can press.
- Display a message and a predefined set of buttons that the operator can respond to.
- Display a keypad so that the operator can enter information, as shown in the example below.

For more information on creating the interactive prompts, see the following command descriptions:

- display.input.number() (on page 8-56)
- display.input.option() (on page 8-58)
- display.input.prompt() (on page 8-59)
- display.input.string() (on page 8-60)

Saving screen captures to a USB flash drive

You can save the content of the front-panel display to a graphic file. The instrument saves these graphic files, also known as screen captures, to the USB flash drive.

To save the screen capture:

1. Insert a USB flash drive in the USB port on front panel of the instrument.
2. Navigate to the screen you want to capture.
3. Press the **HOME** and **ENTER** keys. The instrument displays "Saving screen capture."
4. Release the keys.

Dimensions

The following figures show the mounting screw locations and the dimensions of the instrument with and without the handle and bumpers.

The instrument weighs 4.08 kg (9.0 lb) with the bumpers and handle and 3.63 kg (8.0 lb) without them.

The following figure shows the mounting screw locations and dimensions. Mounting screws must be #6-32 with a maximum screw length of 11.12 mm (0.438 in.) or 7/16 in. The dimensions shown are typical for both sides of the instrument.

Figure 28: Model DMM7510 mounting screw locations and dimensions
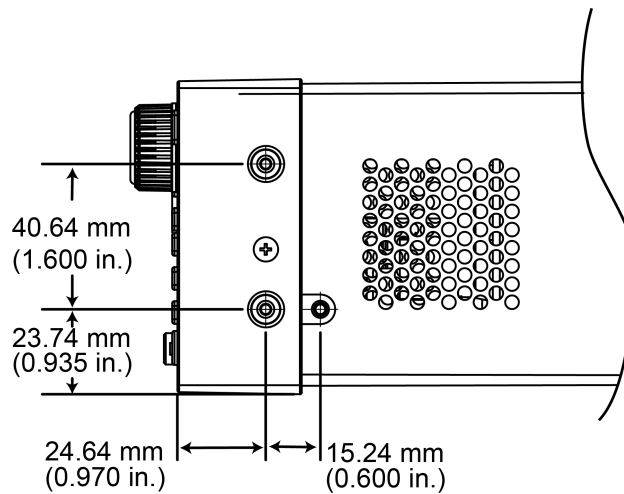
The following figures show the dimensions when the handle and bumpers are installed.

Figure 29: Model DMM7510 dimensions
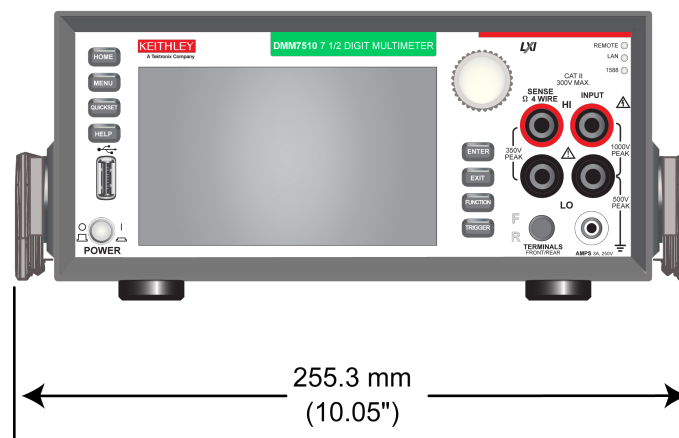
Figure 30: Model DMM7510 dimensions side and top with handle and bumpers
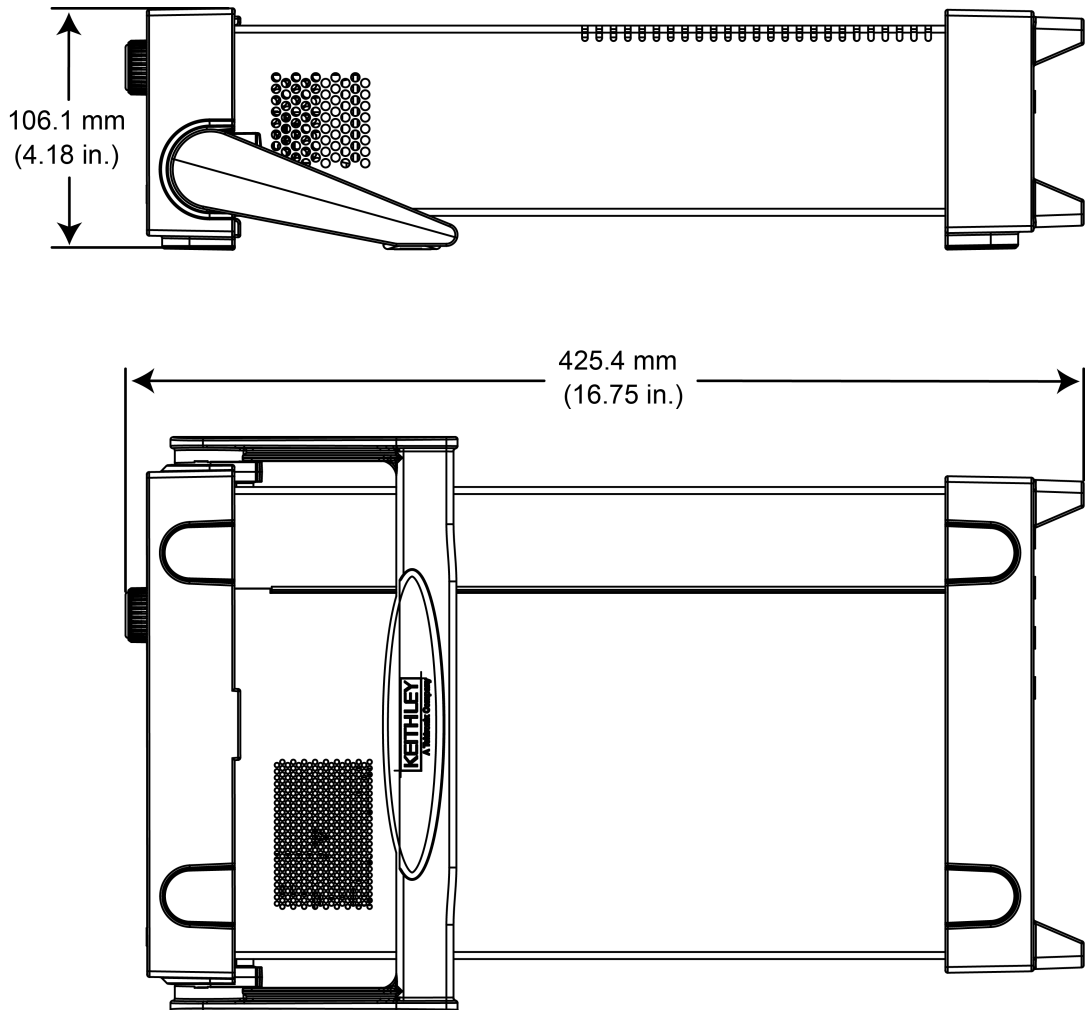
The following figures show the dimensions when the handle and bumpers have been removed.

Figure 31: Model DMM7510 front and rear dimensions (no handle)
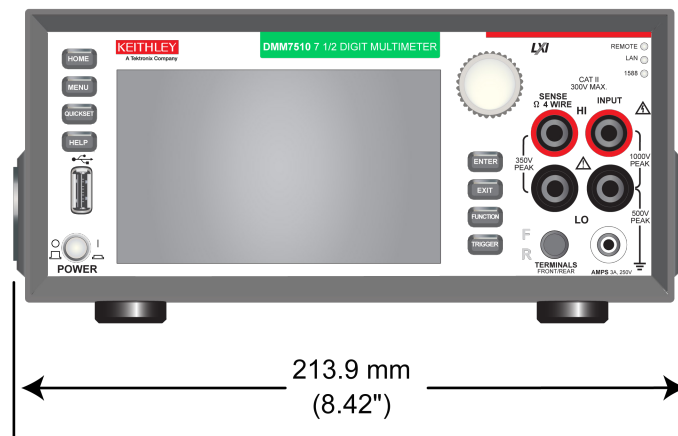
Figure 32: Model DMM7510 top and side dimensions with handle and bumpers removed
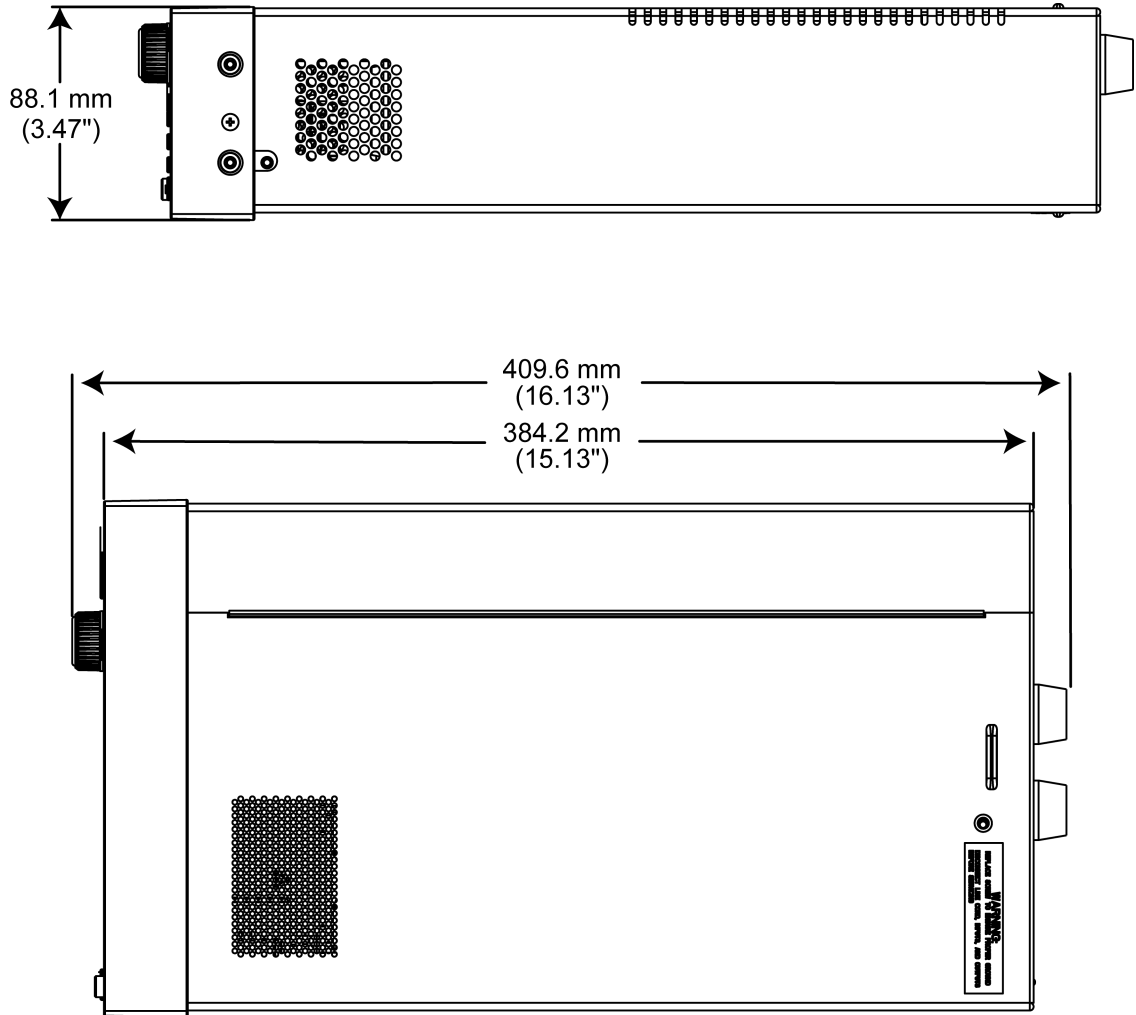
Handle and bumpers

The Model DMM7510 has a handle and front and rear bumpers for using the instrument on a benchtop. The handle rotates so that you can swing it below the bottom surface of the instrument to tilt the instrument up for easier front-panel viewing, or to carry the instrument from one location to another.

Removing the handle and bumpers

You can remove handle and bumpers on the Model DMM7510 if you want to mount the instrument in a rack.

NOTE

If you remove the handle and bumpers, be sure to store them for future benchtop use.

To remove the bumpers:

1. Swivel the handle to a position above or below the instrument so that it will not interfere with the removal of the front bumper.
2. Grasp the front bumper on each side of the Model DMM7510 and gently pull it toward you until the bumper comes off the instrument.

Figure 33: Removing the front bumper
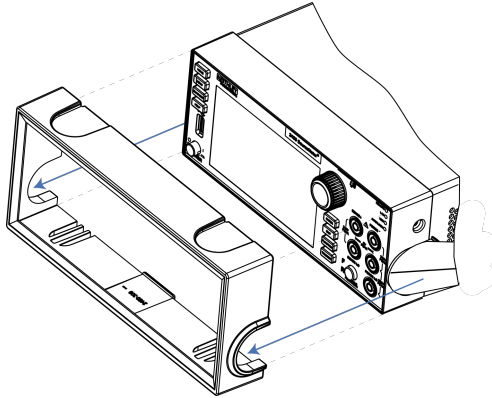
<table>
<tr><td align="center">NOTE</td></tr>
</table>

Remove all connections to the rear panel of the Model DMM7510 before removing the rear bumper.

3. To remove the rear bumper, repeat the procedure in step 2.

To remove the handle assembly:

1. Grasp the sides of the handle near where it attaches to the instrument on both sides and gently pull the handle ends apart to widen the handle as you slide it over the instrument case.

Figure 34: Removing the handle
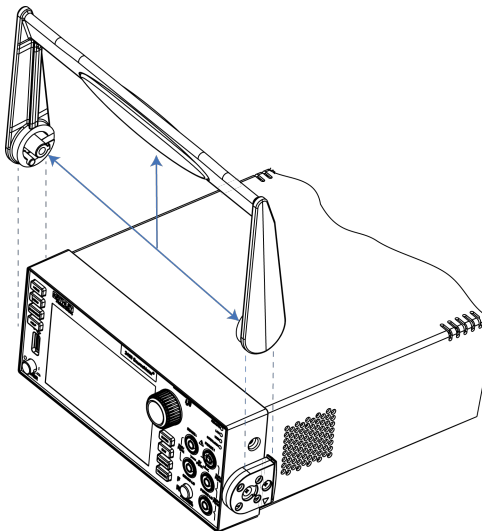
2. Using a Phillips screwdriver, loosen and remove the two screws holding the handle-mount assembly to one side of the Model DMM7510. The handle-mount assembly will fall away from the instrument chassis when the screws are removed.

Figure 35: Removing the handle-mount
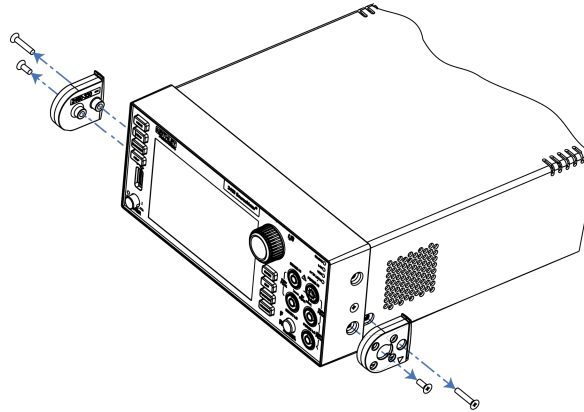
3. Repeat step 2 on the other side of the Model DMM7510.
4. Store the handle-mount assembly, screws, and handle together for future use.

Remote communications interfaces

You can choose from one of several communications interfaces to send commands to and receive responses from the Model DMM7510.

You can control the Model DMM7510 from only one communications interface at a time. The first interface on which it receives a message takes control of the instrument. If another interface sends a message, that interface can take control of the instrument. You may need to enter a password to change the interface, depending on the access mode.

The Model DMM7510 automatically detects the type of communications interface (LAN, GPIB, or USB) when you connect to the respective port on the rear panel of the instrument. In most cases, you do not need to configure anything on the instrument. In addition, you do not need to reboot if you change the type of interface that is connected.

Supported remote interfaces

The Model DMM7510 supports the following remote interfaces:

- **GPIB:** IEEE-488 instrumentation general purpose interface bus
- **USB:** Type B USB port
- **Ethernet:** Local area network ethernet communications
- **TSP-Link:** A high-speed trigger synchronization and communications bus that test system builders can use to connect multiple instruments in a master-and-subordinate configuration

For details about TSP-Link, see TSP-Link System Expansion Interface (on page 3-102).

Comparison of the communications interfaces

The following topics discuss some of the advantages and disadvantages of the communications interfaces that are available for the Model DMM7510.

Simplicity

The GPIB interface is the simplest configuration. Connections are simple, and the only necessary software configuration is setting the instrument address.

An ethernet network is a simple configuration if you can use the automatic settings. It is more complicated if you need to set it up manually. If you must set up your ethernet network manually, you need some knowledge of networking. In addition, your corporate information technology (IT) department may have restrictions that prevent using an ethernet network.

A USB interface is also simple to set up. However, it requires an instrument-specific device driver to communicate with the instrument. This can limit the operating systems that are available for use with the instrument.

Triggering

The GPIB interface provides the fastest, most consistent triggering. It has the lowest trigger latency of the available communications types. Trigger latency is the time that it takes the trigger to go from the computer to the instrument. GPIB also allows you to send triggers to multiple instruments simultaneously.

If you use a USB interface, it is difficult to synchronize triggers that are sent to multiple instruments. For applications that require synchronized triggering, you must use digital I/O. The trigger latency with a USB interface is higher than latency with a GPIB interface, but it is lower and more consistent than latency with an ethernet interface.

Transfer rate

Of the available interfaces, USB has the fastest transfer rate, followed by the ethernet and GPIB interfaces. The GPIB interface, however, offers the most consistent transfer rate.

Instrument naming

Names for instruments that are named through NI-VISA^{TM} are in a human-readable format. USB instrument names are not intended to be human-readable.

Distance and instrument limitations

For GPIB and USB interfaces, the cabling distances between the controller and instrument or hub are limited to 30 feet. In a system connected with GPIB or USB, you can have up to 15 instruments attached to each controller.

The distances for ethernet interfaces are unlimited if the ethernet address of the instrument and ports for the various services it uses are visible publicly (for example, port 80 for web service). If you are using an ethernet interface, you can communicate with an instrument anywhere in the world. In a system that is connected through ethernet, the number of instruments you can attach to each controller is only limited by the controller and the connections available on that controller.

Expense

The GPIB interface is the most expensive method because of the costs for cabling and related equipment. Ethernet and USB connections are inexpensive options because most computers have built-in ethernet and USB ports. In addition, cables and hubs for ethernet and USB interfaces are inexpensive.

GPIB setup

This topic contains information about GPIB standards, bus connections, and primary address selection.

The Model DMM7510 GPIB interface is IEEE Std 488.1 compliant and supports IEEE Std 488.2 common commands and status model topology.

You can have up to 15 devices connected to a GPIB interface, including the controller. The maximum cable length is the lesser of either:

- The number of devices multiplied by 2 m (6.5 ft)
- 20 m (65.6 ft)

You may see erratic bus operation if you ignore these limits.

Install the GPIB driver software

Check the documentation for your GPIB controller for information about where to acquire drivers. Keithley Instruments also recommends that you check the website of the GPIB controller for the latest version of drivers or software.

It is important that you install the drivers before you connect the hardware. This prevents associating the incorrect driver to the hardware.

Install the GPIB cards in your computer

Refer to the documentation from the GPIB controller vendor for information about installing the GPIB controllers.

Connect the GPIB cables to your instrument

To connect an instrument to the GPIB interface, use a cable equipped with standard GPIB connectors, as shown below.

Figure 36: GPIB connector
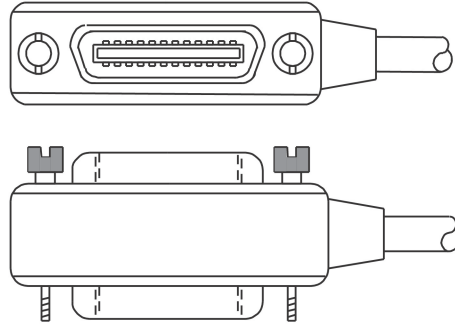
To allow many parallel connections to one instrument, stack the connectors. Each connector has two screws to ensure that connections remain secure. The figure below shows a typical connection diagram for a test system with multiple instruments.

> ⚠ **CAUTION**
>
> To avoid possible mechanical damage, stack no more than three connectors on any one instrument. To minimize interference caused by electromagnetic radiation, use only shielded GPIB cables. Contact Keithley Instruments for shielded cables.

Figure 37: Instrument GPIB connections
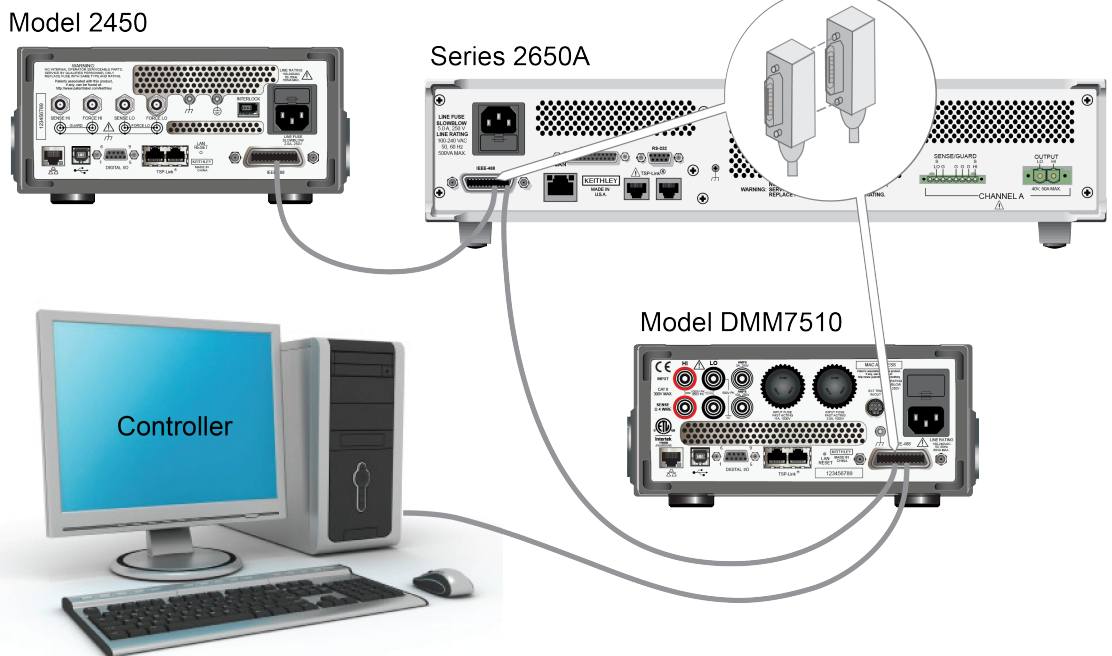
To connect the instrument to the GPIB:

1. Align the cable connector with the connector on the Model DMM7510 rear panel. The location of the connector is shown in the following figure.
2. Attach the connector. Tighten the screws securely but do not overtighten them.

Figure 38: Rear panel GPIB location
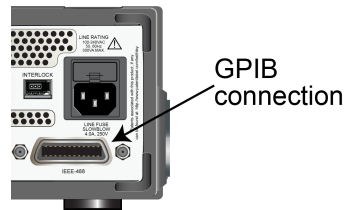
3. Connect any additional connectors from other instruments, as required for your application.
4. Ensure the other end of the cable is properly connected to the controller.

Set the GPIB address

The default GPIB address is 16. You can set the address to any address from 1 to 30 if it is unique in the system. This address cannot conflict with an address that is assigned to another instrument or to the GPIB controller.

<table>
<tr><td>💡 Quick Tip</td></tr>
<tr><td>GPIB controllers are usually set to 0 or 21. To be safe, do not configure any instrument to have an address of 21. To change the controller address, see the documentation for the controller.</td></tr>
</table>

The address is saved in nonvolatile memory, so it does not change when a reset is done or when the power is turned off and then turned on again.

From the front panel:

1. Press the **MENU** key.
2. Under System, select **Communication**. The SYSTEM COMMUNICATION window opens.
3. Select the **GPIB** tab.
4. Next to Address, select the number. The Set GPIB Address dialog box is displayed.
5. Enter the address.
6. Select **OK**.

<table>
<tr><td>NOTE</td></tr>
<tr><td>If you are using a Model DMM7510 with no front panel, you can set the GPIB address with the SCPI command :SYSTem:GPIB:ADDRess (on page 6-140) or TSP command gpib.address (on page 8-214).</td></tr>
</table>

Effect of GPIB line events on Model DMM7510

The GPIB has control lines that allow predefined information, called events, to be transferred quickly. The following information lists some of the GPIB line events and how the Model DMM7510 reacts to them.

DCL

This event clears the GPIB interface. When the Model DMM7510 detects a device clear (DCL) event, it does the following:

* Clears the input buffer, output queue, and command queue
* Cancels deferred commands
* Clears any command that prevents the processing of any other device command

A DCL event does not affect instrument settings and stored data.

GET

The group execute trigger (GET) command is a GPIB trigger that triggers the instrument to take readings from a remote interface.

GTL

When the instrument detects the go to local (GTL) event, it exits remote operation and enters local operation. When the instrument is operating locally, you can control the instrument from the front panel.

IFC

When the instrument detects an interface clear (IFC) event, the instrument enters the talker and the listener idle state. When the instrument is in this state, the GPIB ↑↓ indicators on the front panel are not displayed.

An IFC event does not interrupt the transfer of command messages to and from the instrument. However, messages are suspended. If the transfer of a response message from the instrument is suspended by an IFC event, the transfer resumes when the instrument is addressed to talk. If transfer of a command message to the instrument is suspended by an IFC event, the rest of the message can be sent when the instrument is addressed to listen.

LLO

When the instrument detects a local-lockout (LLO) event, most of the front-panel controls are disabled. This event disables all front-panel controls and POWER switches.

To enable the front panel, use the go-to-local (GTL) event.

REN

When the instrument detects the remote enable (REN) event, it is set up for remote operation. The instrument is not placed in remote mode when it detects the REN event; the instrument must be addressed to listen after the REN event before it goes into remote mode.

You should place the instrument into remote mode before you attempt to program it using a remote interface.

SDC

The selective device clear (SDC) event is similar to the device clear (DCL) event. However, the SDC event clears the interface for an individual instrument instead of clearing the interface of all instruments.

When the Model DMM7510 detects an SDC event, it will do the following for the selected instrument:

- Clears the input buffer, output queue, and command queue
- Cancels deferred commands
- Clears any command that prevents the processing of any other device command

An SDC does not affect instrument settings and stored data.

SPE, SPD

When the instrument detects the serial polling enable (SPE) and serial polling disable (SPD) events, it sends the status byte of the instrument. This contains the serial poll byte of the instrument.

The serial poll byte contains information about internal functions. See the Status model (on page C-1) for detail. Generally, the serial polling sequence is used by the controller to determine which of several instruments has requested service with the SRQ line.

LAN communications

You can communicate with the instrument using a local area network (LAN). The LAN interface can be used to build flexible test systems that include web access. This section provides an overview of LAN communications for the Model DMM7510.

When you connect using a LAN, you can use a web browser to access the internal web page of the instrument and change some of the instrument settings.

The Model DMM7510 is an LXI version 1.4 Core 2011 compliant instrument that supports TCP/IP and complies with IEEE Std 802.3 (ethernet LAN). There is one LAN port (located on the rear panel of the instrument) that supports full connectivity on a 10 Mbps or 100 Mbps network. The Model DMM7510 automatically detects the speed.

The Model DMM7510 also supports Multicast DNS (mDNS) and DNS Service Discovery (DNS-SD), which are useful on a LAN with no central administration.

NOTE

Contact your network administrator to confirm your specific network requirements before setting up a LAN connection.

If you have problems setting up the LAN, refer to LAN troubleshooting suggestions (on page 2-74).

LAN cable connection

The Model DMM7510 includes a Model CA-180-3A cable (LAN crossover cable). You can use this cable for the TSP-Link® network or LAN communications.

However, you can use any standard LAN crossover cable (RJ-45, male to male) or straight-through cable to connect your equipment. The instrument automatically senses which cable you have connected.

The following figure shows the location of the LAN port on the rear panel of the instrument. Connect the LAN cable between this connection and the LAN port on the computer.

Figure 39: Model DMM7510 LAN port

LAN port
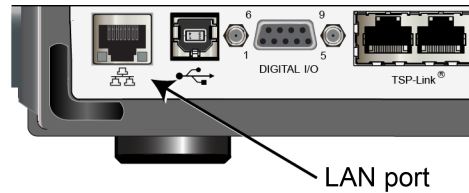
You can connect the instrument to the LAN in a one-to-one, one-to-many, two network cards, or enterprise configuration, as described in the following topics.

One-to-one connection

With most instruments, a one-to-one connection is done only when you are connecting a single instrument to a single network interface card.

A one-to-one connection using a network crossover cable connection is similar to a typical RS-232 hookup using a null modem cable. The crossover cable has its receive (RX) and transmit (TX) lines crossed to allow the receive line input to be connected to the transmit line output on the network interfaces.

Figure 40: One-to-one connection with a crossover cable

CAT5 Crossover

← Up to 100m →
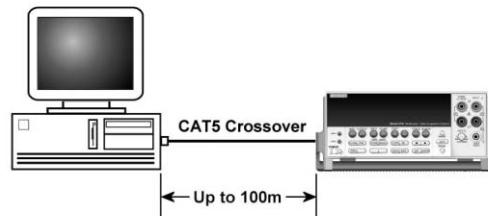
NOTE

The Model DMM7510 supports Auto-MDIX and can use either normal LAN CAT-5 cables (patch) or crossover cables. The instrument automatically adjusts to support either cable.

One-to-many connection

With a LAN hub, a single network interface card can be connected to as many instruments as the hub can support. This requires straight-through network (not crossover) cables for hub connections.

The advantage of this method is easy expansion of measurement channels when the test requirements exceed the capacity of a single instrument. With only the instruments connected to the hub, this is an isolated instrumentation network. However, with a corporate network attached to the hub, the instruments become part of the larger network.

Figure 41: One-to-many connection using a network hub or switch
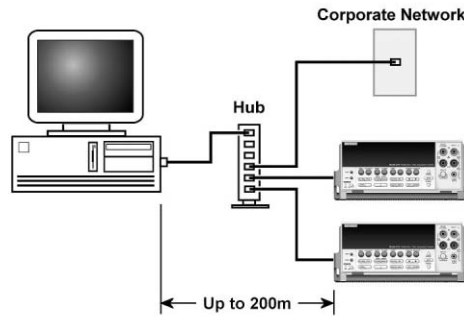
Two network card connection

If you need to connect independent corporate and instrumentation networks, two network interface cards are required in the computer controller. Though the two networks are independent, stations on the corporate network can access the instruments, and the instruments can access the corporate network, using the same computer.

This configuration resembles a GPIB setup in which the computer is connected to a corporate network, but also has a GPIB card in the computer to communicate with instruments.

Figure 42: Two network card connection
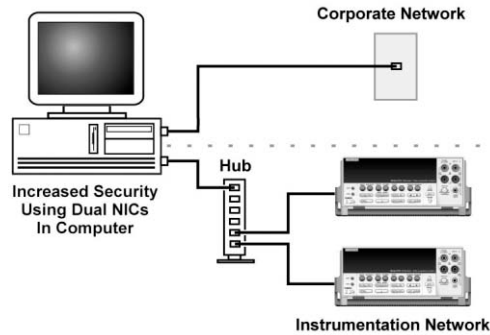
Instrumentation connection to enterprise routers or servers

This connection uses an existing network infrastructure to connect instruments to the computer controller. In this case, you must get the network resources from the network administrator.

Usually, the instruments are kept inside the corporate firewall, but the network administrator can assign resources that allow them to be outside the firewall. This allows instruments to be connected to the Internet using appropriate security methods. Data collection and distribution can be controlled from virtually any location.

Figure 43: Instrumentation connection to enterprise routers or servers
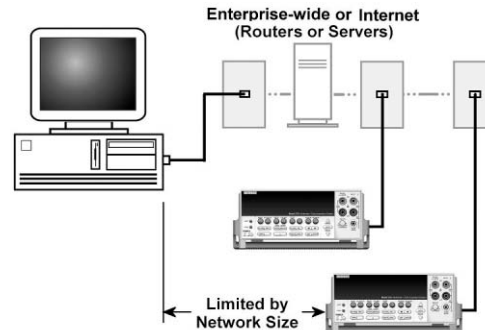
Set up LAN communications on the instrument

This section describes how to set up manual or automatic LAN communications on the instrument.

Check communication settings

Before setting up the LAN configuration, you can check the communications settings on the instrument without making any changes.

To check communications settings on the instrument:

1. Press the **MENU** key.
2. Under System, select **Communication**. The SYSTEM COMMUNICATION window opens.
3. Select one of the four tabs (**GPIB**, **USB**, **LAN**, or **TSP-Link**) to see the settings for that interface.
4. Press the **EXIT** key to leave the SYSTEM COMMUNICATIONS window without making any changes.

NOTE
If you are using a Model DMM7510 with no front panel, you can check the settings with the SCPI command :SYSTem:COMMunication:LAN:CONFigure (on page 6-132) or the TSP command lan.ipconfig() (on page 8-215).

Set up automatic LAN configuration

If you are connecting to a LAN that has a DHCP server or if you have a direct connection between the instrument and a host computer, you can use automatic IP address selection.

If you select Auto, the instrument attempts to get an IP address from a DHCP server. If this fails, it reverts to an IP address in the range of 169.254.1.0 through 169.254.254.255.

NOTE

Both the host computer and the instrument should be set to use automatic LAN configuration. Though it is possible to have one set to manual configuration, it is more complicated to set up.

To set up automatic IP address selection using the front panel:

1. From the home screen, press **MENU**.
2. Under System, select **Communication**.
3. Select the **LAN** tab.
4. For TCP/IP Mode, select **Auto**.
5. Select **Apply Settings** to save your settings.

NOTE

If you are using a Model DMM7510 with no front panel, you can configure the LAN using SCPI or TSP commands. For details, see the SCPI command :SYSTem:COMMunication:LAN:CONFigure (on page 6-132) or the TSP command lan.ipconfig() (on page 8-215).

Set up manual LAN configuration

If necessary, you can set the IP address on the instrument manually.

You can also enable or disable the DNS settings and assign a host name to the DNS server.

NOTE

Contact your corporate information technology (IT) department to secure a valid IP address for the instrument when placing the instrument on a corporate network.

The instrument IP address has leading zeros, but the computer IP address cannot.

To set up manual IP address selection on the instrument:

1. From the home screen, press **MENU**.
2. Under System, select **Communication**.
3. Select the **LAN** tab.
4. For TCP/IP Mode, select **Manual**.
5. For IP Address, enter the LAN IP address. You can touch the number you want to change.
6. For Gateway, enter the gateway address.
7. For Subnet, enter the subnet mask.
8. Select **Apply Settings** to save your settings.

NOTE

If you are using a Model DMM7510 with no front panel, you can configure the LAN using SCPI or TSP commands. For details, see the SCPI command :SYSTem:COMMunication:LAN:CONFigure (on page 6-132) or the TSP command lan.ipconfig() (on page 8-215).

Set up LAN communications on the computer

This section describes how to set up the LAN communications on your computer.

<div style="background:#3b5a8f;color:white;text-align:center;">NOTE</div>

Do not change your IP address without consulting your system administrator. If you enter an incorrect IP address, it can prevent your computer from connecting to your corporate network or it may cause interference with another networked computer.

Record all network configurations before modifying any existing network configuration information on the network interface card. Once the network configuration settings are updated, the previous information is lost. This may cause a problem reconnecting the host computer to a corporate network, particularly if DHCP is disabled.

Be sure to return all settings to their original configuration before reconnecting the host computer to a corporate network. Contact your system administrator for more information.

Wait for the LAN status indicator on the front panel to turn solid green

A solid green LAN status indicator confirms that the instrument was assigned an IP address. Note that it may take several minutes for the computer and instrument to establish a connection.

Install LXI Discovery Browser software on your computer

You can use the LXI Discovery Browser to identify the IP addresses of LXI-certified instruments. Once identified, you can double-click the IP address in the LXI Discovery Browser to open the web interface for the instrument.

The Keithley LXI Discovery Browser is available on the Keithley Instruments website (*http://www.keithley.com*).

To locate the Keithley LXI Discovery Browser on the Keithley website:
1. Select the **Support** tab.
2. In the model number box, type `DMM7510`.
3. From the list, select **Software** and click the search icon. A list of software applications for the instrument is displayed.
4. See the readme file included with the application for more information.

For more information about the LXI Consortium, see the LXI Consortium website (*http://www.lxistandard.org/*).

Run the LXI Discovery Browser

To run the LXI Discovery Browser software:
1. From the Microsoft Windows Start menu, select **Keithley Instruments**.
2. Select **LXI Discovery Browser**.
3. Click **LXI Discovery Browser**. The Keithley LXI Discovery Browser window is displayed.

   The LXI Discovery Browser displays the instruments that it finds on the network and their associated IP addresses.
4. Double-click an IP address in the LXI Discovery Browser dialog box. The instrument web page for that instrument opens.

For information about using the web page, see Model DMM7510 web interface (on page 2-79).

LAN status LEDs

The figure below illustrates the two status light emitting diodes (LED) that are on the LAN port of the instrument. The table below the figure provides explanations of the LED states.

Figure 44: LAN status LEDs
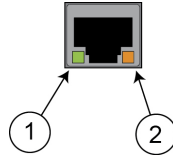
1	When lit, indicates that the LAN port is connected to a 100 Mbps network
2	When blinking, indicates that the port is receiving or sending information

If neither LED is lit, the network is not connected.

LAN interface protocols

You can use one of following LAN protocols to communicate with the Model DMM7510:

- Telnet
- VXI-11
- Raw socket

You can also use a dead socket termination port to troubleshoot communication problems.

NOTE
You can only use one remote interface at a time. Although multiple ethernet connections to the instrument can be opened, only one can be used to control the instrument at a time.

The port numbers for the LAN protocols and dead socket termination are listed in the following table:

LAN protocols

Port number	Protocol
23	Telnet
1024	VXI-11
5025	Raw socket
5030	Dead socket termination

Raw socket connection

All Keithley instruments that have LAN connections support raw socket communication. This means that you can connect to the TCP/IP port on the instrument and send and receive commands. A programmer can easily communicate with the instrument using the Winsock API on computers with the Microsoft® Windows® operating system or using the Berkeley Sockets API on Linux® or Apple® computers.

VXI-11 connection

This remote interface is similar to GPIB and supports message boundaries, serial poll, and service requests (SRQs). A VXI-11 driver or NI-VISA™ software is required. Test Script Builder (TSB) uses NI-VISA and can be used with the VXI-11 interface. You can expect a slower connection with this protocol.

Telnet connection

The Telnet protocol is similar to raw socket, and can be used when you need to interact directly with the instrument. Telnet is often used for debugging and troubleshooting. You will need a separate Telnet program to use this protocol.

The Model DMM7510 supports the Telnet protocol, which you can use over a TCP/IP connection to send commands to the instrument. You can use a Telnet connection to interact with scripts or send real-time commands.

Dead socket connection

The dead socket termination (DST) port is used to terminate all existing ethernet connections. A dead socket is a socket that is held open by the instrument because it has not been properly closed. This most often happens when the host computer is turned off or restarted without first closing the socket. This port cannot be used for command and control functions.

Use the dead socket termination port to manually disconnect a dead session on any open socket. All existing ethernet connections will be terminated and closed when the connection to the dead socket termination port is closed.

Reset LAN settings

You can reset the password and the LAN settings from the rear panel by inserting a straightened paper clip into hole below LAN RESET.

LAN troubleshooting suggestions

If you are unable to connect to the web interface of the instrument, check the following items:

- The network cable is in the LAN port on the rear panel of the instrument, not one of the TSP-Link® ports.
- The network cable is in the correct port on the computer. The LAN port of a laptop may be disabled when the laptop is in a docking station.
- The setup procedure used the configuration information for the correct ethernet card.
- The network card of the computer is enabled.
- The IP address of the instrument is compatible with the IP address on the computer.
- The subnet mask address of the instrument is the same as the subnet mask address of the computer.

You can also try restarting the computer and the instrument. To restart the instrument:

1. Turn the instrument's power off, and then on.
2. Wait at least 60 seconds for the network configuration to be completed.
3. Press the **MENU** key.
4. Under System, select **Communication**.
5. Select the **LAN** tab.
6. Verify the settings.

If the above actions do not correct the problem, contact your system administrator.

USB communications

To use the rear-panel USB port, you must have the Virtual Instrument Software Architecture (VISA) layer on the host computer. See How to install the Keithley I/O Layer (on page 2-84) for more information.

VISA contains a USB-class driver for the USB Test and Measurement Class (USBTMC) protocol that, once installed, allows the Microsoft® Windows® operating system to recognize the instrument.

When you connect a USB device that implements the USBTMC or USBTMC-USB488 protocol to the computer, the VISA driver automatically detects the device. Note that the VISA driver only automatically recognizes USBTMC and USBTMC-USB488 devices. It does not recognize other USB devices, such as printers, scanners, and storage devices.

In this section, "USB instruments" refers to devices that implement the USBTMC or USBTMC-USB488 protocol.

NOTE

The full version of National Instruments (NI®) VISA provides a utility to create a USB driver for any other kind of USB device that you want to communicate with VISA. For more information, see the National Instruments (*http://www.ni.com*) website.

Using USB

A USB cable is shipped with the instrument. If the original cable is not available, you will need a USB cable with a USB Type B connector end and a USB type A connector end. You will need a separate USB cable for each instrument you plan to connect to the computer at the same time using the USB interface.

To use a USB connection:

1. Connect the Type A end of the cable to the host computer.
2. Connect the Type B end of the cable to the instrument.
3. Turn power to the instrument on.
4. When the host computer detects the new USB connection, the Found New Hardware Wizard starts.
5. On the "Can Windows connect to Windows Update to search for software?" dialog box, click **No**, and then click **Next**.
6. On the "USB Test and Measurement device" dialog box, click **Next**, and then click **Finish**.

Communicate with the instrument

For the instrument to communicate with the USB device, you must use NI-VISA™. VISA requires a resource string in the following format to connect to the correct USB instrument:

`USB0::0x05e6::0x7510::[serial number]::INSTR`

Where:

- `0x05e6`: The Keithley vendor ID
- `0x7510`: The instrument model number
- `[serial number]`: The serial number of the instrument (the serial number is also on the rear panel)
- `INSTR`: Use the USBTMC protocol

To determine these parameters, you can run the Keithley Configuration Panel, which automatically detects all instruments connected to the computer.

If you installed the Keithley I/O Layer, you can access the Keithley Configuration Panel through the Microsoft® Windows® Start menu.

To use the Keithley Configuration Panel to determine the VISA resource string:

1. Click **Start > All Programs > Keithley Instruments > Keithley Configuration Panel**. The Select Operation dialog box is displayed.

Figure 45: Select Operation dialog box
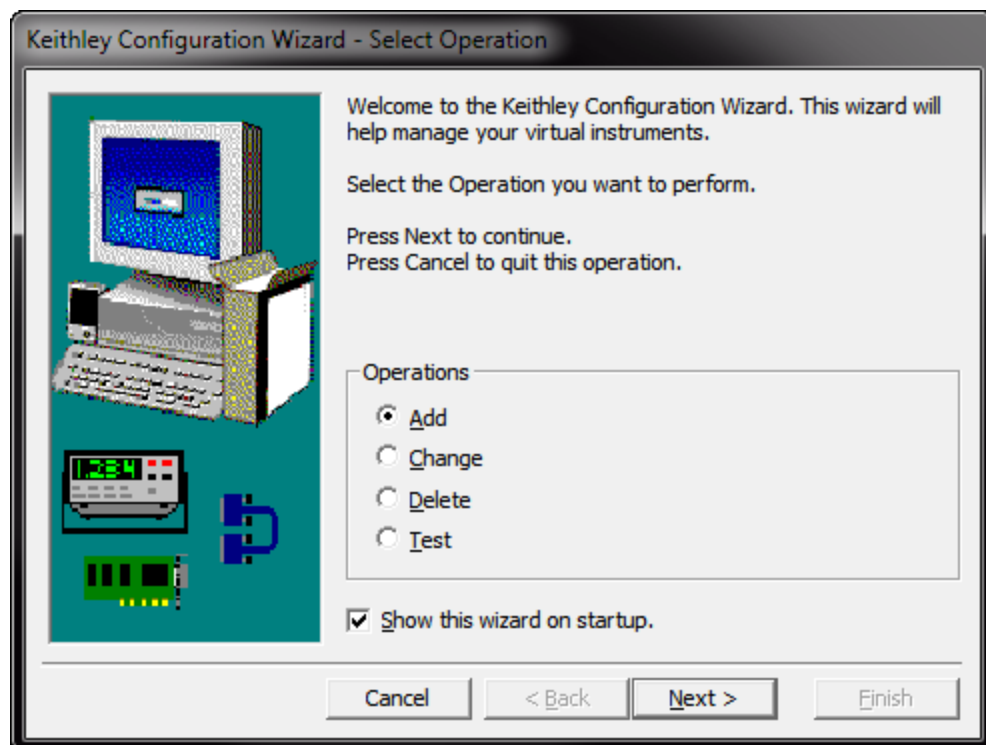
2. Select **Add**.

3. Click **Next**. The Select Communication Bus dialog box is displayed.

Figure 46: Select Communication Bus dialog box
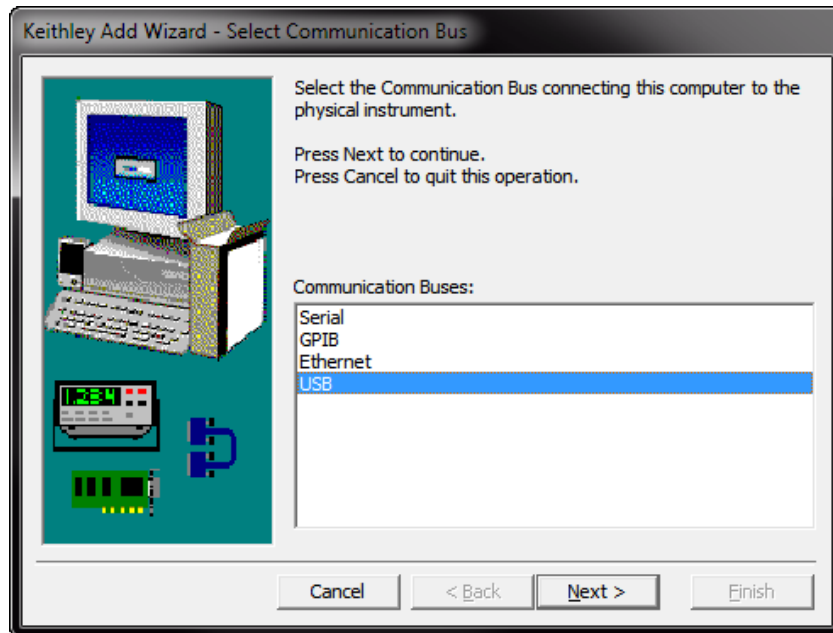
4. Select **USB**.
5. Click **Next**. The Select Instrument Driver dialog box is displayed.

Figure 47: Select Instrument Driver dialog box
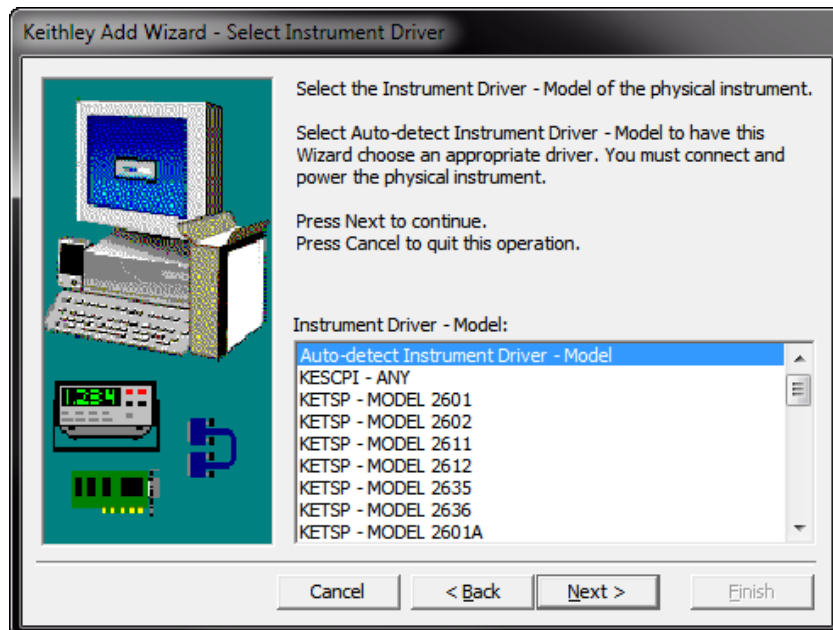
6. Select **Auto-detect Instrument Driver - Model**.
7. Click **Next**. The Configure USB Instrument dialog box is displayed with the detected instrument VISA resource string visible.
8. Click **Next**. The Name Virtual Instrument dialog box is displayed.

Figure 48: Name Virtual Instrument dialog box
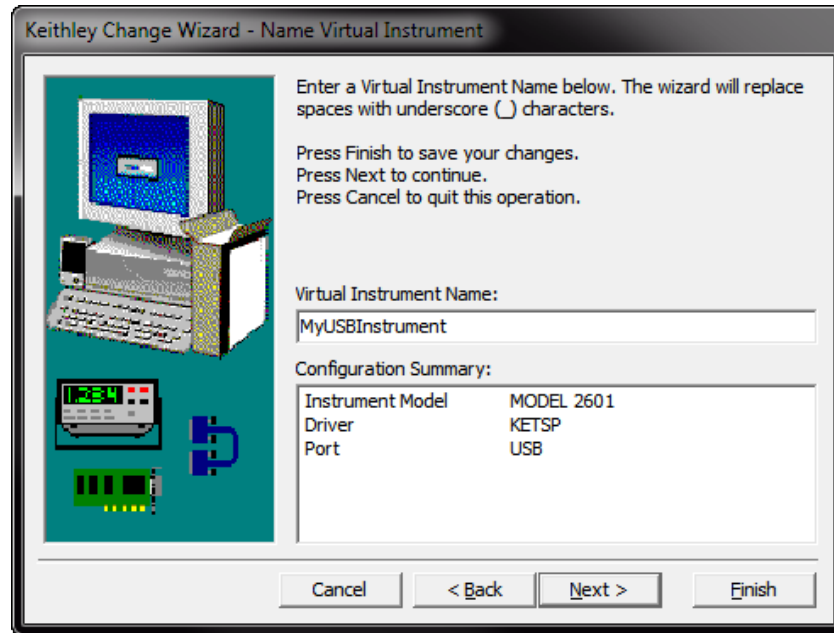
9. In the Virtual Instrument Name box, enter a name that you want to use to refer to the instrument.
10. Click **Finish**.
11. Click **Cancel** to close the Wizard.
12. Save the configuration. From the Keithley Configuration Panel, select **File > Save**.

Verify the instrument through the Keithley Communicator:

1. Click **Start > All Programs > Keithley Instruments > Keithley Communicator**.
2. Select **File > Open Instrument** to open the instrument you just named.

Figure 49: Keithley Communicator Open an Instrument
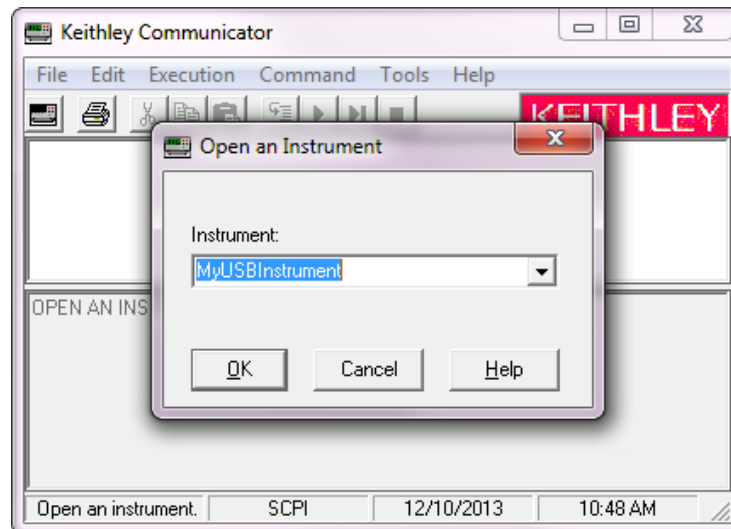
3. Click **OK**.
4. Send a command to the instrument and see if it responds.

NOTE

If you have a full version of NI-VISA on your system, you can run NI-MAX or the VISA Interactive Control utility. See the National Instruments documentation for information.

If you have the Agilent IO Libraries on your system, you can run Agilent Connection Expert to check your USB instruments. See the Agilent documentation for information.

Model DMM7510 web interface

The Model DMM7510 web interface allows you to make settings and control your instrument through a web page. The web page includes:

- Instrument status.
- The instrument model, serial number, firmware revision, and the last LXI message.
- An ID button to help you locate the instrument.
- A virtual front panel and command interface that you can use to control the instrument.
- Download access to a .csv file that contains reading buffer data.
- Administrative options and LXI information.

The instrument web page resides in the firmware of the instrument. Changes you make through the web interface are immediately made in the instrument.

When the LAN and instrument establish a connection, you can open a web page for the instrument.

To access the web interface:

1. Open a web browser on the host computer.
2. Enter the IP address of the instrument in the address box of the web browser. For example, if the instrument IP address is 192.168.1.101, enter `192.168.1.101` in the browser address box.
3. Press **Enter** on the computer keyboard to open the instrument web page.
4. If prompted, enter a user name and password. The default is `admin` for both.

NOTE

If the web page does not open in the browser, see LAN troubleshooting suggestions (on page 2-74).

♀ Quick Tip

To find the IP Address of the instrument, press the Communications indicator in the upper left corner of the Home screen.

Figure 50: Model DMM7510 web-interface home page
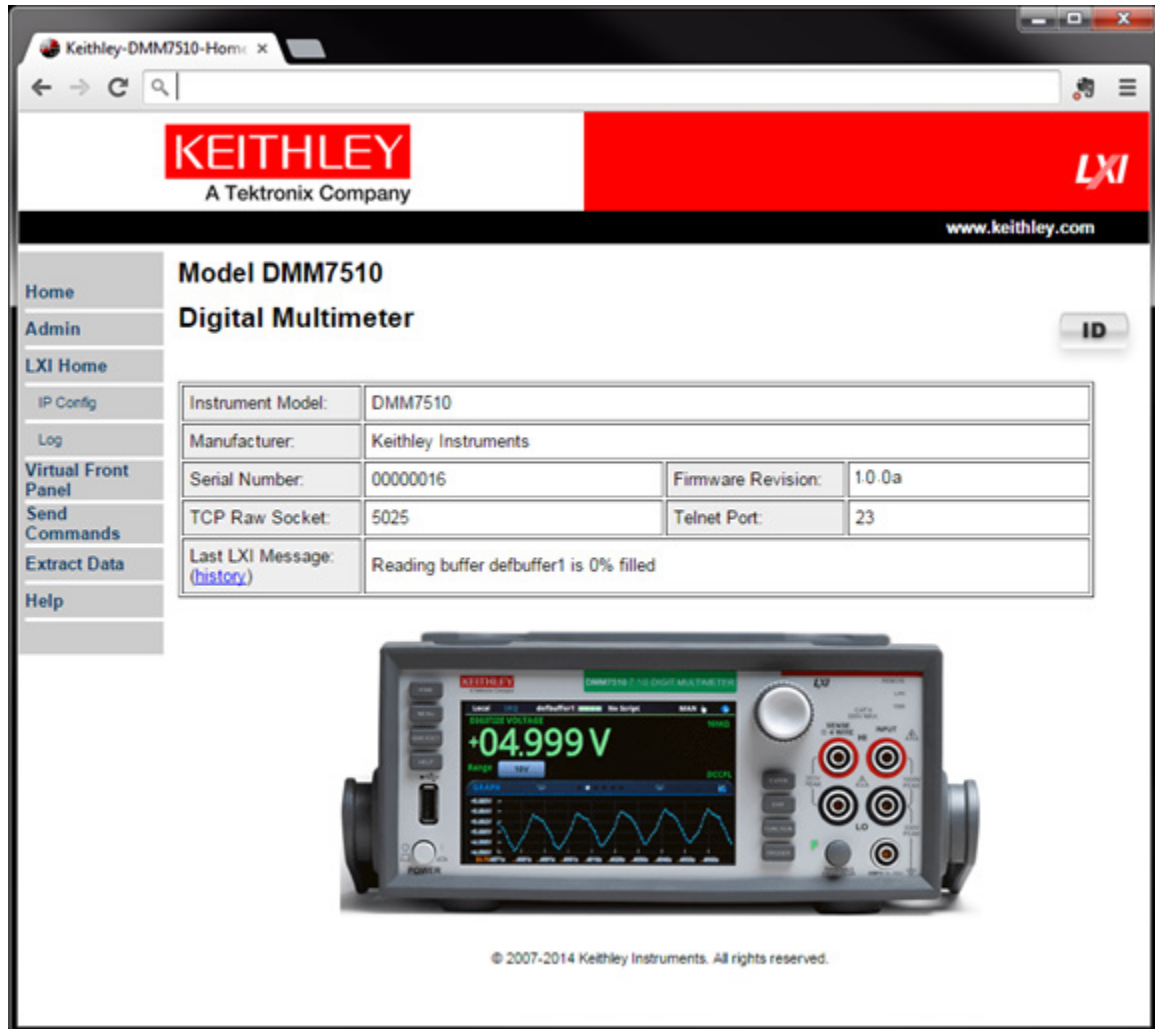
The Home page of the instrument provides information about the instrument. It includes:

- The instrument model number, manufacturer, serial number, and firmware revision number.
- The TCP Raw Socket number and Telnet Port number.
- The last LXI message. The history link opens the LXI Home page (on page 2-81).
- The ID button, which allows you to identify the instrument. Refer to Identify the instrument (on page 2-81).

Identify the instrument

If you have a bank of instruments, you can click the ID button to determine which one you are communicating with.

To identify the instrument:

In the upper right corner of the Home page, click the **ID** button.

The button turns green and the LAN status indicator on the instrument blinks.

Click the **ID** button again to return the button to its original color and return the LAN status indicator to steady on.

LXI Home page

The LXI Home page displays instrument information, including the host name, MAC address, and VISA resource string. You cannot change the information from this page.

You can use the host name instead of the IP address to connect to the instrument.

It also includes the ID button, which you can use to identify the instrument. See Identify the instrument (on page 2-81).

Change the IP configuration through the web interface

You can change the LAN settings, such as IP address, subnet mask, gateway, and DNS address, through the web page of the instrument.

If you change the IP address through the web page, the web page will try to redirect to the IP address that is configured in the instrument. In some cases, this may fail. This generally happens if you switch from IP address assignment that uses a static address to IP address assignment that uses a DHCP server. If this happens, you need to revert to either using the front panel to set the IP address or use an automatic discovery tool to determine the new IP address.

NOTE
You can also change the IP configuration through the front panel or with TSP and SCPI commands. See Set up LAN communications on the instrument (on page 2-70) for information.

To change the IP configuration using the instrument web page:

1. Access the internal web page as described in Connecting to the instrument through the web interface (on page 2-79).
2. From the navigation bar on the left, in the LXI Home menu, select **IP Config**.
3. Click **Modify**. The Modify IP Configuration page is displayed.

Hostname:	K-7510-00000000
Description:	Keithley DMM7510 #00000000
TCP/IP Configuration Mode:	○ Automatic ⦿ Manual
Static IP Address:	134.63.71.199
Subnet Mask:	255.255.255.192
Default Gateway:	134.63.71.193
DNS Server:	134.63.75.12
Domain:	

Submit

4. Change the values.

5. Click **Submit**. The instrument reconfigures its settings, which may take a few moments.

> ## NOTE
> You may lose your connection with the web interface after clicking **Submit**. This is normal and does not indicate an error or failure of the operation. If this occurs, find the correct IP address and reopen the web page of the instrument to continue.

Review events in the event log

The event log records all LXI events that the instrument generates and receives. The log includes the following information:

- The EventID column, which shows the event identifier that generated the event.
- The System Timestamp column, which displays the seconds and nanoseconds when the event occurred.
- The Data column, which displays the text of the event message.

To clear the event log and update the information on the screen, click the **Refresh** button.

Using the Model DMM7510 virtual front panel

The Virtual Front Panel page allows you to control the instrument from a computer as if you were using the front panel. You can operate the instrument using a mouse to select options.

The virtual front panel operates the same as the actual front panel, with the following exceptions:

- The navigation control cannot be turned.
- The Front/Rear Terminals button only indicates the setting of the switch. You cannot change which set of terminals is used remotely.
- To scroll up or down on a screen, hold the left mouse button down and swipe up or down.
- To scroll right or left, hold the left mouse button down and swipe left or right. You can also click the dots on the bar above the swipe screens to move from screen to screen.
- You cannot use pinch and zoom on the graph screen.

Pause Updates allows you to stop updates from the instrument. You can use this to freeze data on the screen. Click **Resume Updates** to start updating again.

To use the virtual front panel, you can use any of the standard web browsers. If you are using Microsoft Internet Explorer, it must be version 9 or above. Earlier versions will not allow the swipe motion to work.

> ## NOTE
> Using graphing through the virtual front panel requires significant system resources and may slow instrument operation.

For information on the options, see <u>Screen descriptions</u> (on page 2-12).

See the following figure.

Figure 51: Model DMM7510 virtual front panel
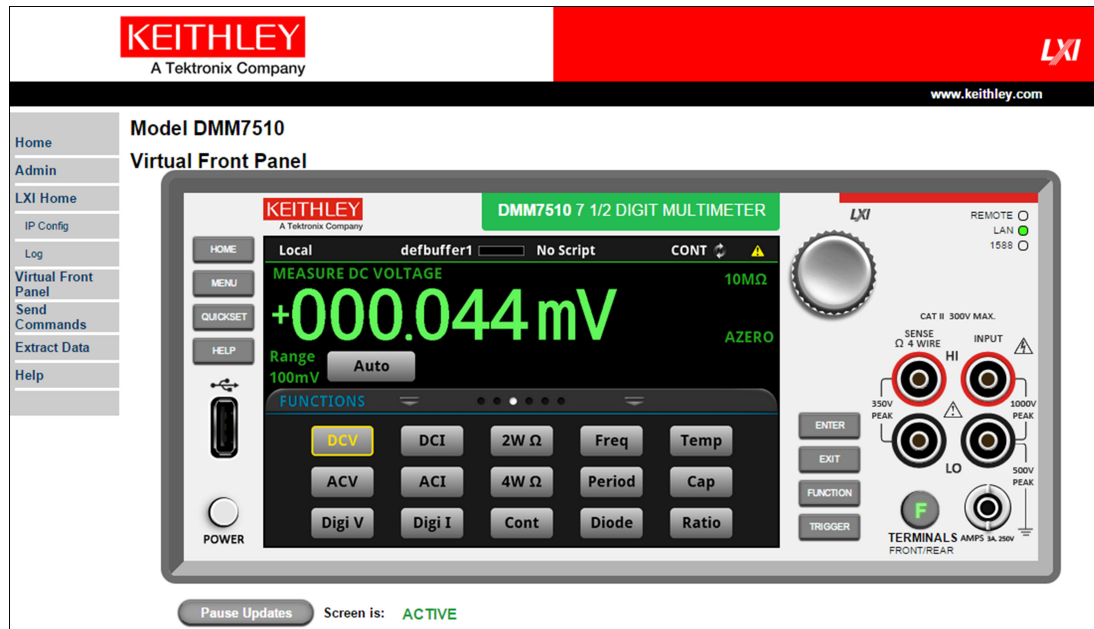
> **NOTE**
>
> The Model DMM7510 only allows fewer than three clients to open the virtual front panel web page at the same time. Only the first successfully connected client can operate the instrument. Other clients can only view the virtual front panel.

Change the date and time through the web interface

You can change the instrument date and time through the web interface. This is the same as changing the date and time through the front panel System Settings menu. The date and time is used for the event log entries and data timestamps.

To change the date and time:
1. From the web interface page, select **Admin**.
2. In the Local Time table, change the information as needed.
3. Click **Submit**.

Change the password through the web interface

You can change the instrument password from the web interface.

The default user name and password is `admin`. Note that you cannot change the user name; it remains at `admin` even if the password has changed.

To change the password:
1. From the web interface Home page, select **Admin**.
2. In the **Current password** box, enter the presently used password.
3. In the **New password** and **Confirm new password** boxes, enter the new password.
4. Click **Submit.**

Send commands using the web interface

You can send individual commands using the web interface.

The active command set is listed above the Command box.

To send commands using the web page:

1. From the navigation bar on the left, click **Send Commands**.
2. If requested, log in.
3. In the **Command** box, enter the command.
4. Click **Send Command** to send the command to the instrument. The command is displayed in the Command Output box. If there is a response to the command, it is displayed after the command.
5. To view any events that have occurred, click **Return Error**.
6. To clear the Command Output list, click **Clear Output**.

Extract buffer data using the web interface

The Extract Data page of the web interface allows you to download reading buffer data from the instrument.

To download buffer data:

1. From the web interface page, click **Extract Data**.
2. In the CSV File column, click the name of the file that you want to download.
3. Follow the instructions for your browser to open the file. Typically, the file will open in Microsoft Excel.

How to install the Keithley I/O Layer

NOTE
Before installing, it is a good idea to check the Keithley Instruments website (*http://www.keithley.com*) to see if a later version of the Keithley I/O Layer is available. On the website, select the **Support** tab, under **model number**, type `KIOL`, and select **Software Driver**.

You can install the Keithley I/O Layer from the CD-ROM that came with your instrument, or from the download from the Keithley website.

The software installs the following components:

- Microsoft® .NET Framework
- NI™ IVI Compliance Package
- NI-VISA™ Run-Time Engine
- Keithley SCPI-based Instrument IVI-C driver
- Keithley I/O Layer

To install the Keithley I/O Layer from the CD-ROM:

1. Close all programs.
2. Place the CD-ROM into your CD-ROM drive.
3. Your web browser should start automatically and display a screen with software installation links. If you need to manually open the web page, use a file explorer to navigate to the CD-ROM drive and open the file named `index.html`.
4. From the web page, select the **Software** category and click **Keithley I/O Layer**.
5. Accept all defaults.
6. Click **Next**.
7. Click **Install**.
8. Turn your computer off and then on again to complete the installation.

To install the Keithley I/O Layer from the Keithley website:

1. Download the Keithley I/O Layer Software from the Keithley Instruments website (*http://www.keithley.com*) as described in the note. The software is a single compressed file and should be downloaded to a temporary directory.
2. Run the downloaded file from the temporary directory.
3. Follow the instructions on the screen to install the software.
4. Reboot your computer.

Modifying, repairing, or removing Keithley I/O Layer software

The Keithley I/O Layer interconnects many other installers.

To remove all the KIOL components, you need to uninstall the following applications using Control Panel:

- National Instruments NI^{TM} IVI Compliance Package
- National Instruments NI-VISA^{TM} Run-Time Engine
- IVI Shared Components
- Visa Shared Components
- Keithley SCPI Driver

After uninstalling components, reboot the computer.

Determining the command set you will use

You can change the command set that you use with the Model DMM7510. The remote command sets that are available include:

- SCPI: An instrument-specific language built on the SCPI standard.
- TSP: A scripting programming language that contains instrument-specific control commands that can be executed from a stand-alone instrument. You can use TSP to send individual commands or use it to combine commands into scripts.

You cannot combine the command sets.

NOTE

As delivered from Keithley Instruments, the Model DMM7510 is set to work with the Model DMM7510 SCPI command set.

To set the command set from the front panel:
1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Select the button next to Command Set.
4. Select the command set.
5. You are prompted to reboot.

To verify which command set is selected from a remote interface:

Send the command:

```
*LANG?
```

To change to the SCPI command set from a remote interface:

Send the command:

```
*LANG SCPI
```

Reboot the instrument.

To change to the TSP command set from a remote interface:

Send the command:

```
*LANG TSP
```

Reboot the instrument.

System information

You can get the serial number, firmware build, detected line frequency, calibration verify date, calibration adjust date, and calibration adjust count information from the instrument.

To view the version and serial number information from the front panel:
1. Press the **MENU** key.
2. Under System, select **Info/Manage**.

The firmware version and serial number are displayed at the top of the page.

To view the calibration information from the front panel:

1. Press the **MENU** key.
2. Under System, select **Calibration**.

The adjust date, adjust count, and calibration date are displayed at the bottom of the page.

To view the line frequency information from the front panel:

1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Scroll down to display the line frequency at the bottom of the page.

To view system information using SCPI commands:

To retrieve the manufacturer, model number, serial number, and firmware version, send the command:

```
*IDN?
```

To read the line frequency, send the command:

```
SYStem:LFRequency?
```

The firmware build, memory available, and calibration date are not available when using SCPI commands.

To view system information using TSP commands:

To read the model number, send the command:

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

To read the serial number, send the command:

```
print(localnode.serialno)
```

To read the firmware version, send the command:

```
print(localnode.version)
```

To read the line frequency, send the command

```
print(localnode.linefreq)
```

The calibration date is not available with TSP commands.

You can also create user-defined strings to store custom, instrument-specific information in the instrument, such as department number, asset number, or manufacturing plant location. See the TSP command reference (on page 8-1) for detail about the `userstring` functions.

Instrument sounds

The instrument can emit a beep when a front-panel key is pressed or when a system event occurs. You can turn these beeps on or off.

Through the remote interface, you can generate a beep with a defined length and tone. This is typically used as part of code to indicate that something has occurred.

With front-panel access:

To turn off beeps when system events occur (setting is only available from the front panel):
1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Next to Audible Errors, select **On** or **Off**.

To turn the key clicks on or off (setting is only available from the front panel):
1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Next to Key Click, select **On** or **Off**.

To generate an audible tone from the SCPI remote interface:
```
:SYSTem:BEEPer <frequency, time>
```

Where `frequency` is the frequency of the sound in Hz (20 to 20,000) and `time` is the length of the sound in seconds.

To generate an audible tone from the TSP command interface:

Send the following command:
```
beeper.beep(duration, frequency)
```

Where *duration* is the length of the sound in seconds and *frequency* is the frequency of the sound in Hz (20 to 20,000).

Test connections

> ## ⚠ WARNING
>
> *To prevent electric shock, test connections must be configured such that the user cannot come in contact with test leads or any device under test (DUT) that is in contact with the conductors. It is good practice to disconnect DUTs from the instrument before powering the instrument. Safe installation requires proper shields, barriers, and grounding to prevent contact with test leads.*
>
> *There is no internal connection between protective earth (safety ground) and the LO terminals of the Model DMM7510. Therefore, hazardous voltages (more than 30 V$_{rms}$) can appear on LO terminals. This can occur when the instrument is operating in any mode. To prevent hazardous voltage from appearing on the LO terminals, connect the LO terminal to protective earth if your application allows it. You can connect the LO terminal to the chassis ground terminal on the front panel or the chassis ground screw terminal on the rear panel. Note that the front-panel terminals are isolated from the rear-panel terminals. Therefore, if you are using the front-panel terminals, ground to the front-panel LO terminal. If using the rear-panel terminals, ground to the rear panel LO terminal.*
>
> *Be aware that hazardous voltages can appear on the LO terminals even if the terminals are not presently selected. The TERMINALS FRONT/REAR switch selects the active terminals for the measurement. It does not disconnect the terminals.*
>
> *The maximum input voltage between INPUT HI and INPUT LO is 1000 V peak. Exceeding this value may create a shock hazard.*
>
> *The maximum common-mode voltage (the voltage between INPUT LO and chassis ground) is 500 V peak. Exceeding this value may cause a breakdown in insulation that can create a shock hazard.*

You can make test connections to the Model DMM7510 from the rear or front panel of the instrument.

Basic connections

You can access the INPUT HI, INPUT LO, SENSE LO, and SENSE HI connections from the front or rear panel of the instrument. The connections are banana jacks.

The front and rear panels of the instrument show the maximum allowable voltage differentials between terminals. The maximum common-mode voltage is the voltage between INPUT LO and ground. You must limit the current from an external common-mode voltage source. You can use protective impedance or a fuse to limit the current.

When making or breaking connections, follow these guidelines:

- Power off the Model DMM7510 and all other instruments.

- Disconnect any devices that may deliver energy.

- Make connections to the device under test through a test fixture or other safe enclosure.

- Make sure the Model DMM7510 is properly connected to protective earth (safety ground).

- If the test fixture is conductive, make sure the test fixture is properly connected to protective earth (safety ground).

- Make sure the test fixture provides proper protection.

- Properly make interlock connections between the Model DMM7510, the test fixture, and any other instruments.

- Make sure to follow all warnings and cautions and to take adequate safety precautions for each set of connections.

- Properly terminate any triaxial cables. All unterminated cable ends must be in a safe enclosure.

- See Two-wire local sense connections and Four-wire remote sense connections for examples of connections.

Front- or rear-panel test connections

You can use either the front-panel or the rear-panel terminals to make connections to the device under test (DUT). The instrument must be set to use either the front or the rear terminals.

NOTE
You cannot make some connections to the front-panel terminals and some to the rear-panel terminals for the same test setup. All connections for the same test must be made to either the front-panel or the rear-panel terminals.

⚠ WARNING
Be aware that hazardous voltages can appear on the LO terminals even if the terminals are not presently selected. The TERMINALS FRONT/REAR switch selects the active terminals for the measurement. It does not disconnect the terminals.

Determining whether to use front or rear terminals

Both front and rear terminals are banana-jack connectors to the device under test.

The rear terminals allow for current up to 10 A. The front-panel allow for current up to only 3 A.

Otherwise, you can make your DUT connections from either the front or rear panel based on convenience. For example, the front connections may work better for benchtop applications that require frequent connection changes. The rear connections may work better for rack applications with fewer changes.

Setting the instrument to use the front or rear terminals

The selection to use the front or rear terminals must be made using the front-panel switch. There are no remote commands that can be used to set the terminals.

Using the front panel:

1. Press the **TERMINALS FRONT/REAR** switch.

When F is lit, the instrument reads from the front-panel terminals. When R is lit, the instrument reads from the rear-panel terminals.

DMM measurement overview

This section describes the connections and basics of making the measurements for each function.

<table>
<tr><td align="center">NOTE</td></tr>
<tr><td>The measurement overview presented here assumes that the measurement method is set to Continuous Measurement (the default). Select the trigger mode indicator to change the measurement method to Continuous, if necessary.</td></tr>
</table>

Figure 52: Trigger mode indicator

DMM measurement capabilities

The Model DMM7510 can make the following measurements:

- DC voltage measurements from 10 nV to 1000 V
- AC true RMS voltage measurements from 0.1 μV to 700 V
- DC current measurements from 1 pA to 10 A
- AC current measurements from 1 nA to 10 A
- 2-wire resistance measurements from 0.1 $\mu\Omega$ to 1 GΩ
- 4-wire resistance measurements from 0.1 $\mu\Omega$ to 1 GΩ
- Continuity measurements from 100 mΩ to 1 kΩ
- Frequency measurements up to a minimum of 1 MHz on voltage signals from 100 mV to 700 V
- Period measurements up to a minimum of 1 MHz on voltage signals from 100 mV to 700 V
- Diode measurements from 1 μV to 10 V
- 3-wire and 4-wire RTD measurements from -200 C to 630 C
- Thermistor measurements from -80 C to 150 C
- Capacitance measurements from 0.1 pF to 1000 μF
- Vinput and Vsense measurements from 10 nV to 1000 V; the Vsense measurements are only available on the 100 mV, 1 V, and 10 V ranges
- Digitize voltage measurements from 10 μV to 1000 V
- Digitize current measurements from 1 nA to 10 A

Warmup time

After the Model DMM7510 is turned on, it must be allowed to warm up for at least 1½ hours to allow the internal temperature to stabilize. If the instrument has been exposed to extreme temperatures, allow extra warmup time.

High-energy circuit safety precautions

To optimize safety when measuring voltage in high-energy distribution circuits, read and use the directions in the following warning.

⚠ WARNING

Dangerous arcs of an explosive nature in a high-energy circuit can cause severe personal injury or death. If the Model DMM7510 is connected to a high-energy circuit when set to a current range or low resistance range, the circuit is virtually shorted. Dangerous arcing can result even when the Model DMM7510 is set to a voltage range if the minimum voltage spacing is reduced in the external connections.

As described in International Electrotechnical Commission (IEC) Standard IEC 60664, the Model DMM7510 measurement ports are rated CAT II up to 300 V. This range must not be exceeded. In addition, the Model DMM7510 must not be connected to CAT III or CAT IV circuits. Failure to comply can result in instrument damage or severe personal injury.

When making measurements in high-energy circuits, use test leads that meet the following requirements:

- Test leads should be fully insulated.
- Only use test leads that can be connected to the circuit (for example, alligator clips and spade lugs) for hands-off measurements.
- Do not use test leads that decrease voltage spacing. These diminish arc protection and create a hazardous condition.

Power circuit test procedure

Use the following procedure when testing power circuits:

1. Turn off power to the circuit using the regular installed connect-disconnect device. For example, remove the device's power cord or turn off the power switch.
2. Attach the test leads to the circuit under test. Use appropriate safety-rated test leads for this application. If over 42 V, use double-insulated test leads or add an additional insulation barrier for the operator.
3. Set the Model DMM7510 to the proper function and range.
4. Power the circuit using the installed connect-disconnect device, and make measurements without disconnecting the multimeter.
5. Remove power from the circuit using the installed connect-disconnect device.
6. Disconnect the test leads from the circuit under test.

DC voltage measurements

This section describes how you can set up DC voltage measurements from the front panel.

⚠ CAUTION

Inputs: Do not apply more than 1000 V_{peak} between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

DC voltage measure connections

Figure 53: Front panel connections: DC voltage measurement
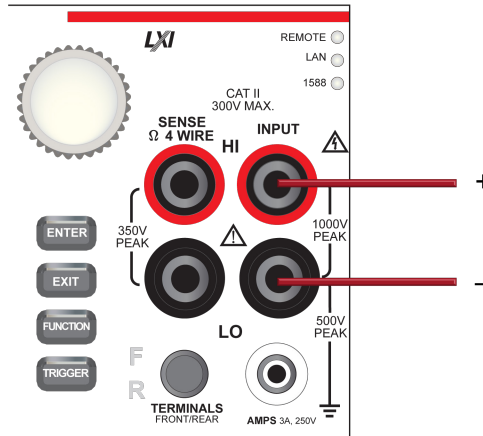
Figure 54: Rear panel connections: DC voltage measurement
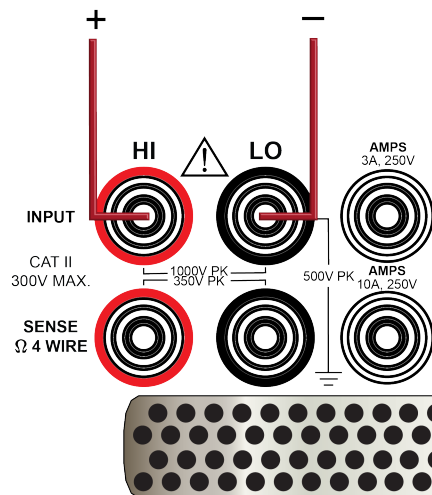
Measure DC voltage using the front panel

To make a DC voltage measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **DC Voltage**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for DC voltage measurements

See DC voltage measure settings (on page 2-24) for the settings that are available when you are making DC voltage measurements.

Show voltage readings in decibels

You can show DC or AC voltage in decibels (dB), which compresses a large range of measurements into a much smaller scope. The relationship between dB and voltage is defined by the following equation:

$$dB = 20 \log \left| \frac{V_{in}}{V_{ref}} \right|$$

Where:

V_{in} is the DC or AC input signal

V_{ref} is the specified voltage reference level

If a relative offset value is in effect when dB is selected, the value is converted to dB, and then relative offset is applied to the dB value. If relative offset is applied after dB has been selected, dB has relative offset applied to it.

NOTE

The largest negative value of dB is −160 dB. This accommodates a ratio of V_{in} = 1 µV and V_{ref} = 1000 V.

DC voltage input impedance

You can set the input impedance for the DC voltage and digitize voltage functions to automatic (AUTO) or 10 MΩ for all ranges.

Automatic input impedance provides the lowest measure noise with the highest isolation on the device under test (DUT). When automatic input impedance is selected, the 100 mV to 10 V voltage ranges have more than 10 GΩ input impedance. For the 100 V and 1000 V ranges, a 10 MΩ input divider is placed across the HI and LO input terminals.

When the input impedance is set to 10 MΩ, the 100 mV to 1000 V ranges have a 10 MΩ input divider across the HI and LO input terminals. The 10 M impedance provides stable measurements when the terminals are open (approximately 100 µV at 1 PLC).

Choosing automatic input impedance is a balance between achieving low DC voltage noise on the 100 mV and 1 V ranges and optimizing measurement noise due to charge injection. The Model DMM7510 is optimized for low noise and charge injection when the DUT has less than 100 KΩ input resistance. When the DUT input impedance is more than 100 K, selecting an input impedance of 10 MΩ optimizes the measurement for lowest noise on the 100 mV and 1 V ranges. You can achieve short-term low noise and low charge injection on the 100 mV and 1 V ranges with autozero off. For the 10 V to 1000 V ranges, both input impedance settings achieve low charge injection.

When you enable the 10 MΩ input divider, the measurement INPUT HI is connected to INPUT LO.

Note that when the input divider is enabled, some external devices (such as high-voltage probes) must be terminated to a 10 MΩ load.

Setting input impedance from the front panel:
1. Press the **MENU** key.
2. Select **Settings**.
3. Select the **Input Impedance** setting.

Setting input impedance using SCPI commands:

Refer to [:SENSe[1]]:<function>:INPutimpedance (on page 6-79).

Setting input impedance using TSP commands:

For the DC voltage function, refer to dmm.measure.inputimpedance (on page 8-152).

For the digitize voltage function, refer to dmm.digitize.inputimpedance (on page 8-86).

AC voltage measurements

This section describes how you can set up AC voltage measurements from the front panel.

⚠ CAUTION

Do not apply more than 1000 V$_{peak}$ between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

AC voltage measure connections

Figure 55: Front panel connections: AC voltage measurement
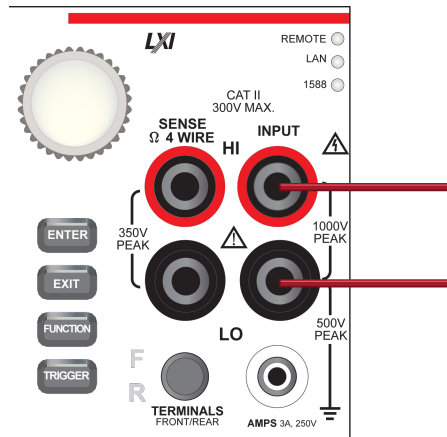
Figure 56: Rear panel connections: AC voltage measurement
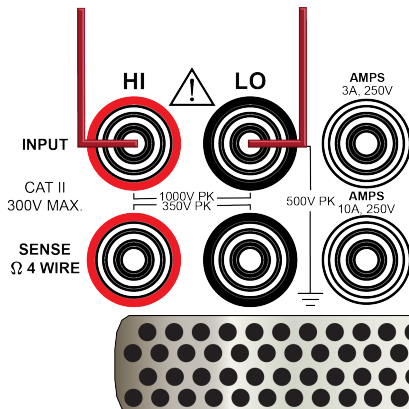
Measure AC voltage using the front panel

To make an AC voltage measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **AC Voltage**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for AC voltage measurements

See AC voltage measure settings (on page 2-25) for settings that are available when you are making AC voltage measurements.

DC current measurements

This section describes how you can set up DC current measurements from the front panel.

⚠ WARNING
To prevent electric shock, never make or break connections while power is present in the test circuit.

DC current measure connections

Figure 57: Front panel connections: DC current measurement (3 A or less)
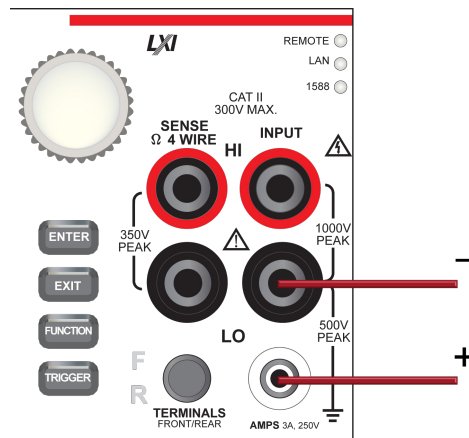
Figure 58: Rear panel connections: DC current measurement (3 A or less)
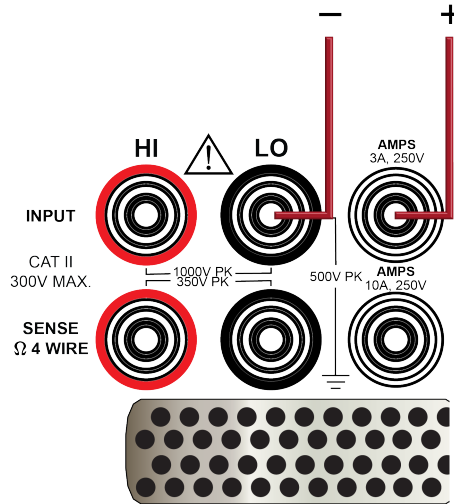
Figure 59: Rear panel connections: DC current measurement (10 A or less)
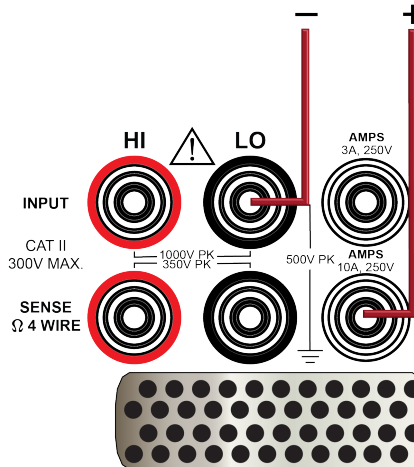
Measure DC current from the front panel

To make a DC current measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **DC Current**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

NOTE
When the TERMINALS switch is set to REAR and autorange is enabled, autoranging is limited to ranges up to 3 A ranges. The 10 A range is not included in the autorange algorithm.

Settings available for DC current measurements

See DC current measure settings (on page 2-26) for settings that are available when you are making DC current measurements.

AC current measurements

This section describes how you can set up AC current measurements from the front panel.

⚠ WARNING
To prevent electric shock, never make or break connections while power is present in the test circuit.

AC current measure connections

Figure 60: Front-panel connections: AC current measurement (3 A or less)
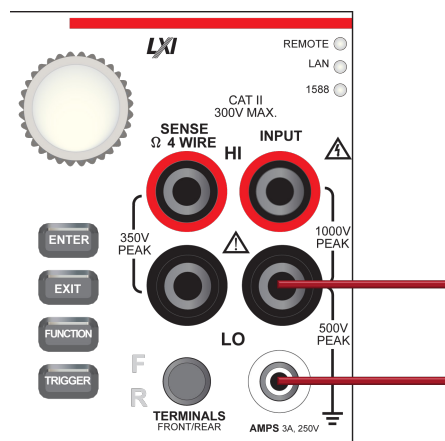
Figure 61: Rear-panel connections: AC current measurement (3 A or less)
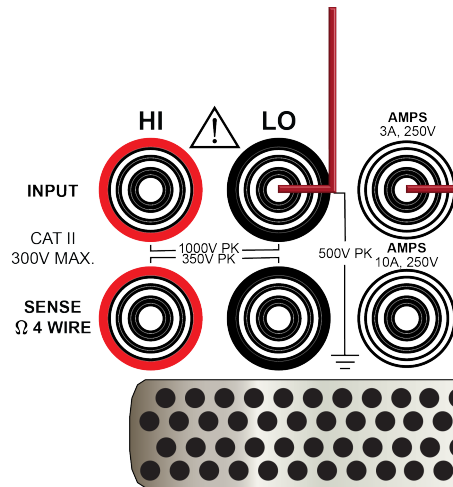
Figure 62: Rear-panel connections: AC current measurement (3 A or more)
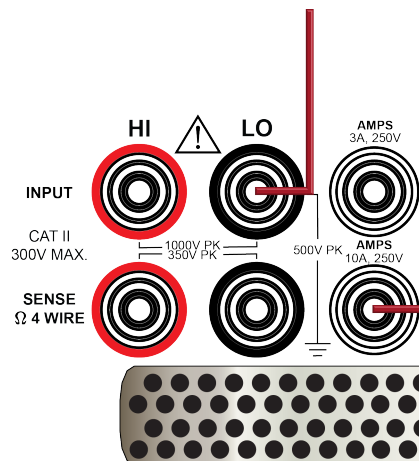
Measure AC current using the front panel

To make an AC current measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **AC Current**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

NOTE

When the TERMINALS switch is set to REAR and autorange is enabled, autoranging is limited to ranges up to 3 A ranges. The 10 A range is not included in the autorange algorithm.

Settings available for AC current measurements

See AC current measure settings (on page 2-26) for settings that are available when you are making AC current measurements.

Resistance measurements

You can make 2-wire or 4-wire resistance measurements with the Model DMM7510.

For resistances more than 1 kΩ, the two-wire method is typically used for measurements. For resistances less than 1 kΩ, use the 4-wire measurement method to cancel the effect of test-lead resistance.

⚠ CAUTION

Do not apply more than 1000 V$_{peak}$ between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

For high resistance measurements in a high humidity environment, use Teflon^{TM} insulated cables to minimize errors due to cable leakage.

Two-wire compared to four-wire measurements

You can use 2-wire or 4-wire measurement techniques with the Model DMM7510.

You should use 4-wire, or remote sense, measurement techniques for the following conditions:

- Low impedance applications
- When measuring impedance that is less than 100 KΩ

Use 4-wire connections when you are concerned about voltage drops because of lead or contact resistance that could affect measurement accuracy. This can occur on low-impedance devices when you are measuring through a relay switch card and the channel on resistance.

You can use the 2-wire, or local sense, measurement technique for the following measure conditions when the voltage drop due to the 2-wire test current and cable lead resistance is minimal compared to the resistance of the device under test.

Accuracy of 2-wire resistance measurements

The 2-wire sensing method has the advantage of requiring only two test leads and provides faster reading rates. However, as shown in the following figure, the total lead resistance is added to the measurement. This can seriously affect the accuracy of 2-wire resistance measurements, particularly with low resistance values.

Figure 63: Two-wire resistance sensing for high impedance DUT
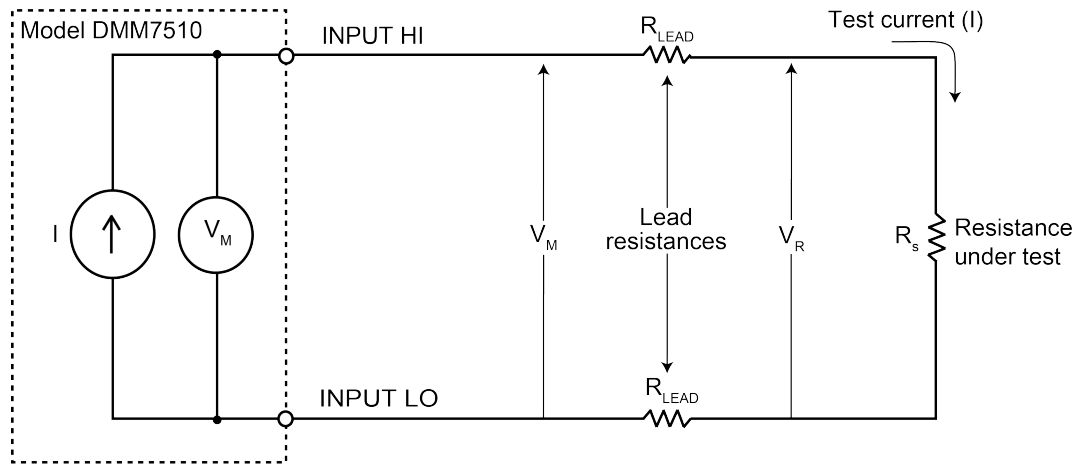
$$\text{Measured resistance} = \frac{V_M}{I} = R_s + (2 \times R_{LEAD})$$

$$\text{Actual resistance} = \frac{V_R}{I} = R_s$$

I = Test current
V_M = Voltage measured
V_R = Voltage across resistor

Minimizing the effect of lead resistance with 4-wire testing

The 4-wire sensing method, shown in the following figure, minimizes or eliminates the effects of lead resistance. The effects of lead resistance are minimized by measuring the voltage across the resistor under test with a second set of test leads. The current through the sense leads is negligible, and the measured voltage is essentially the same as the voltage across the resistor under test. Note that the voltage-sensing leads should be connected as close to the resistor under test as possible to avoid including the resistance of the test leads in the measurement.

Figure 64: Model DMM7510 4-wire resistance sensing.png
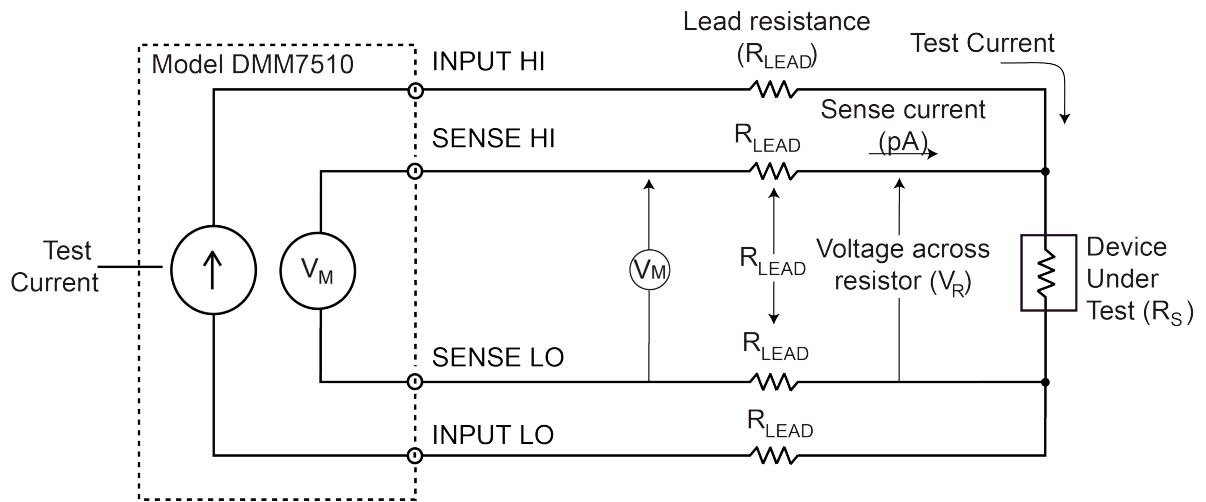
Sense current is negligible, therefore $V_M = V_R$ Measure resistance is $\dfrac{V_M}{I} = \dfrac{V_R}{I} = R_s$

Open lead detection

When 4-wire measurements are made, erratic readings can occur if the Sense HI, Sense LO, or both terminals are open. This can be caused by broken test leads.

To prevent erratic readings from open leads, you can enable the open lead detector feature. When open lead detection is enabled and the range is 1 Ω to 1 MΩ ranges, the instrument pulses a 1 ms negative current on the Sense HI and Sense LO terminals. If the signal at either terminal is less than -10 mV, the display reads "OverflowΩ". If the signal is more than -10 mV, the current pulse is automatically shut off, and the 4-wire measurement continues. For the 10 MΩ to 1 GΩ ranges, only the Sense LO terminal is pulsed with a negative current, which minimizes settling time and device-under-test noise.

When open lead detection is enabled, there is minimal impact on reading rates and an increase in measurement reliability and integrity.

Open lead detection reduces the reading rate by 2 ms while Sense HI and Sense LO are measured. Also, for measurements made through long capacitive cables or switch cards, the open lead detection pulse current can increase settling time and decrease accuracies, especially for the 10 kΩ to 1 MΩ ohm ranges.

Two-wire local sense connections

Two-wire connections are shown in the following figures.

Figure 65: Two-wire DUT connections to rear panel
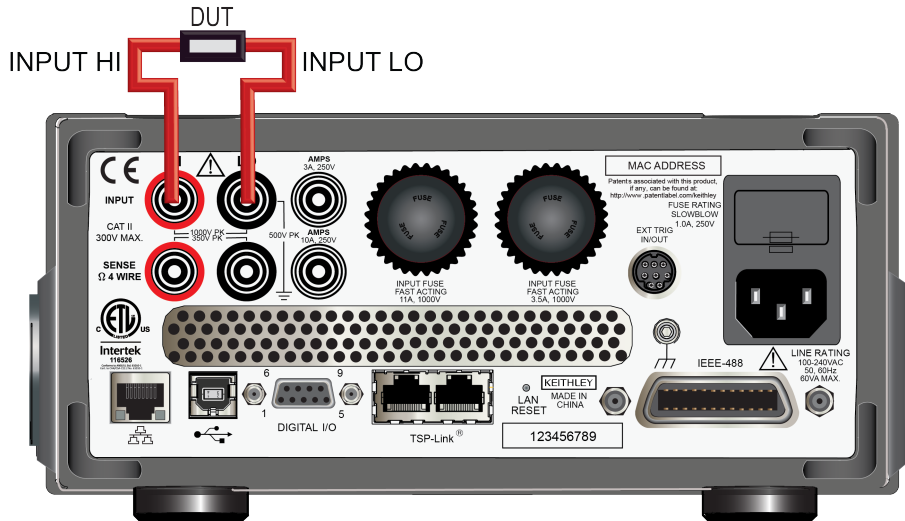
Figure 66: Two-wire DUT connections to the front panel
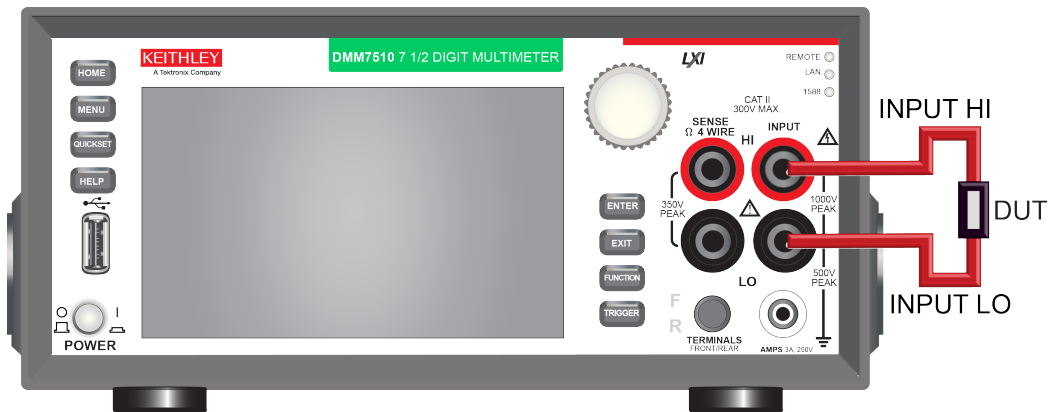
Four-wire remote sense connections

Using 4-wire remote sense connections provides the most accurate low resistance measurement accuracy. Specified accuracies for instrument measurement capabilities are only guaranteed when you use 4-wire remote sensing.

Four-wire connections are shown in the following figures.

Four-wire remote sense connection drawings

Always connect the sense lines as close as possible to the device under test.

Figure 67: Model DMM7510 rear-panel 4-wire remote sense connections
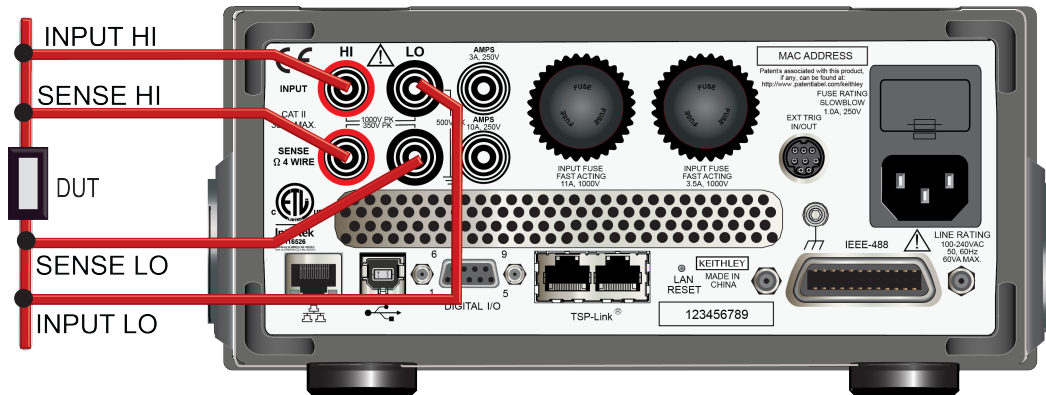
Figure 68: Model DMM7510 front-panel 4-wire remote sense connections
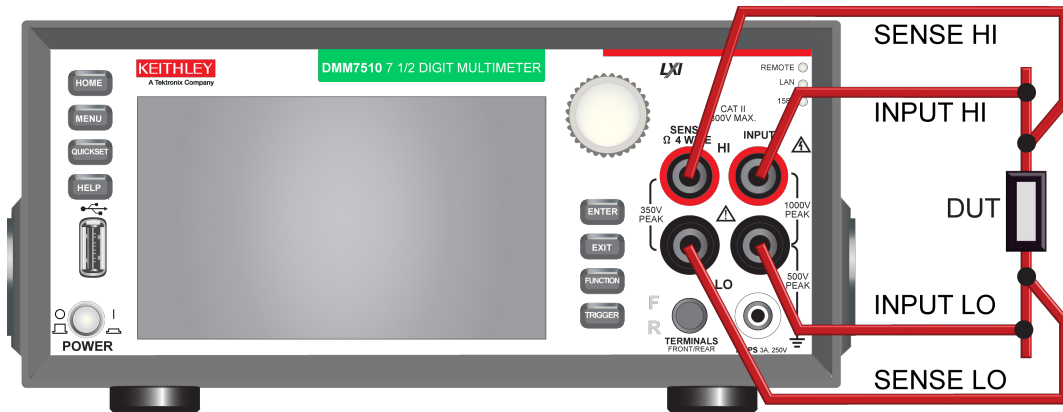
2-wire resistance measure connections

Figure 69: Front panel connections: 2-wire resistance measurement
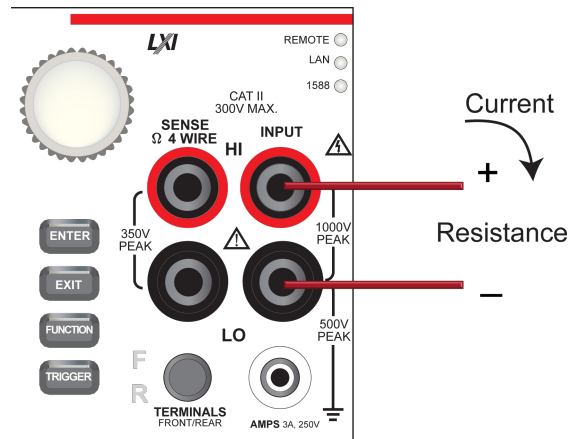
Figure 70: Rear panel connections: 2-wire resistance measurement
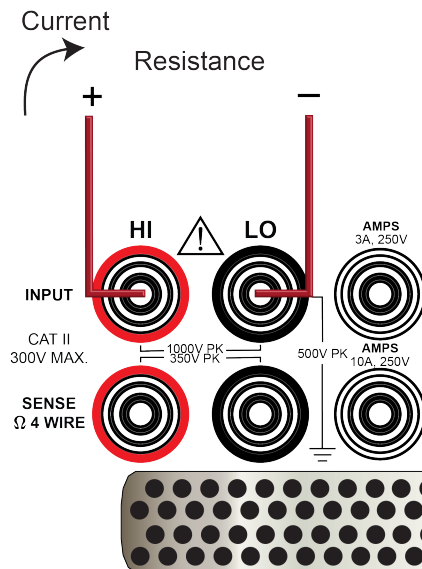
Measure 2-wire resistance using the front panel

To make a 2-wire resistance measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **2W Res**.
4. If the measurement method is set to continuous, the measurements start displaying on the front-panel display.

    If the measurement method is set to manual or trigger model, press the **TRIGGER** key to make a measurement.

Settings available for 2-wire resistance measurements

See 2-wire resistance measure settings (on page 2-27) for settings that are available when you are making 2-wire resistance measurements.

4-wire resistance measure connections

Figure 71: Front panel connections: 4-wire resistance measurement
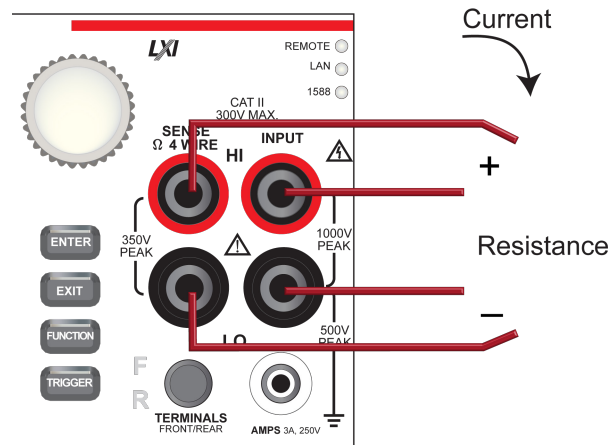
Figure 72: Rear panel connections: 4-wire resistance measurement
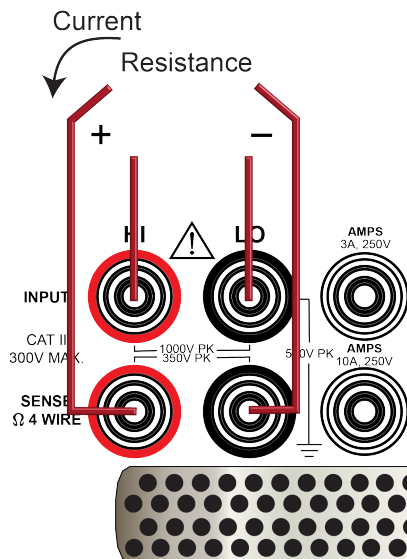
Measure 4-wire resistance using the front panel

To make a 4-wire resistance measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **4W Res**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for 4-wire resistance measurements

See 4-wire resistance measure settings (on page 2-28) for settings that are available when you are making 4-wire resistance measurements.

Offset-compensated ohms

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

For 4-wire resistance measurements, when offset compensation is enabled, the measure range is limited to a maximum of 100 kΩ. Offset compensation is automatically enabled when dry circuit is enabled.

For 2-wire resistance measurements, offset compensation is always set to off.

For temperature measurements, offset compensation is only available when the transducer type is set to 3-wire or 4-wire RTD.

See Offset-compensated ohm calculations (on page 4-15) for additional detail on calculating offset-compensated ohms.

Dry circuit ohms

Standard resistance measurements have open-circuit voltage levels from 6.4 V to 14.7 V, depending on the selected range. Dry circuit ohms limits open-circuit voltage to between 20 mV and 27 mV. This allows you to perform resistance measurements that require low open-circuit voltage, such as power and low-glitch resistance measurements.

You can use dry circuit ohms for ranges up to 10 kΩ (maximum resistance of 2.4 kΩ) for the four-wire resistance function.

You can use offset-compensated ohms used with dry circuit ohms to cancel the effect of thermoelectric EMFs. When dry circuit is enabled, offset compensation is automatically set to on.

Measuring contact resistance (oxide film build-up)

The ideal resistance between switch connectors or relay contacts is 0 Ω. However, an oxide film may be present on the switch or relay contacts. This oxide film could add resistance on the order of several hundred milliohms. Also, this oxide film changes the contact resistance over time and with changes in the environmental conditions (such as temperature and humidity).

Typically, the four-wire ohm function of the Model DMM7510 or a standard DMM is used to measure low resistance. However, if standard resistance measurements are performed, the relatively high open-circuit voltage may puncture the oxide film, and render the test meaningless.

Dry circuit ohms limit voltage to 25 mV to minimize any physical and electrical changes in a measured contact junction. This low open-circuit voltage will not puncture the film, and will therefore provide a resistance measurement that includes the resistance of the oxide film.

Oxide films may also build up in connections on a semiconductor wafer. To accurately measure the resistance introduced by the oxide film, dry circuit ohms should be used to prevent oxide film puncture.

Enabling or disabling dry circuit ohms

Dry circuit ohms is only available for the 4-wire resistance function.

When the dry circuit ohms feature is enabled, DRYCR is displayed to the right of the measurement on the front panel of the instrument.

With 4-wire measurement 1 to 100 kΩ ranges and dry circuit 1 to 10 kΩ ranges, thermal voltages in the test leads or device under test can create measure errors. To eliminate the thermal offset, offset compensation is automatically enabled when you enable dry circuit ohms.

When offset compensation is enabled, two measurement phases are made. The first is with the I_{off} test current, which is sourced from HI and LO and measured across the Sense HI and Sense LO terminals. The second measurement is measured across Sense HI and Sense LO with an I_{on} test current from HI and LO. The difference between the I_{on} and I_{off} measurements is applied as an offset to eliminate any thermal voltages.

NOTE

When the dry circuit ohms attribute is enabled, the offset-compensated ohms attribute is automatically enabled (OCMP displayed to the right of the measurement). If you do not wish to use offset-compensated ohms, after setting dry circuit ohms, disable offset-compensated ohms.

From the front panel:

1. Press the **FUNCTION** key.
2. Select **4W Res**.
3. Press the **MENU** key.
4. Swipe to scroll down.
5. Set Dry Circuit to **ON** to enable or **OFF** to disable.
6. Press the **HOME** key to return to the measurement display.

Using SCPI commands:

Send the commands:

```
:SENSe:FUNCtion "FRES"
:SENSe:FRES:DCIRcuit ON
```

Using TSP commands:

Send the commands:

```
dmm.measure.func = dmm.FUNC_4W_RESISTANCE
dmm.measure.drycircuit = dmm.ON
```

Measuring dry circuit ohms

Make sure you use four-wire connections to the device under test. Refer to 4-wire resistance measure connections (on page 2-107).

> ## NOTE
>
> Do not make connections to the device under test (DUT) until after the dry circuit ohms feature is set to on.

To measure dry circuit ohms from the front panel:

1. Enable dry circuit ohms. Refer to Enabling or disabling dry circuit ohms (on page 2-109).
2. Make 4-wire connections to the DUT.
3. Select the trigger method annunciator and select **Manual Trigger Mode**.
4. Press the **TRIGGER** key.
5. Observe the displayed reading. If the "Overflow" message is displayed, select a higher range until a normal reading is displayed, or select Auto to use autoranging. If you are selecting a manual range, use the lowest possible range for the best resolution.

> ## NOTE
>
> As with other measure settings, the states of dry circuit ohms and offset-compensated ohms are saved with four-wire ohm function. If you select a different measurement function, then select four-wire ohms again, the previous on or off states of dry circuit ohms and offset-compensated ohms are restored.

Continuity measurements

This section describes how you can set up continuity measurements from the front panel.

The Model DMM7510 can test continuity using the 2-wire 1 kΩ range with a user-selected threshold resistance level. When the measured circuit is below the set threshold level, the instrument displays the resistance readings. When the measured circuit is above the threshold level, the instrument displays the message "OPEN."

The continuity function does not support relative offset. Use the mx+b (on page 3-7) calculation, with b as an offset, to compensate for cable resistance.

> ## NOTE
>
> The reading rate for continuity is always set to 0.006 power line cycles.

Continuity measure connections

Figure 73: Front panel connections: Continuity measurement
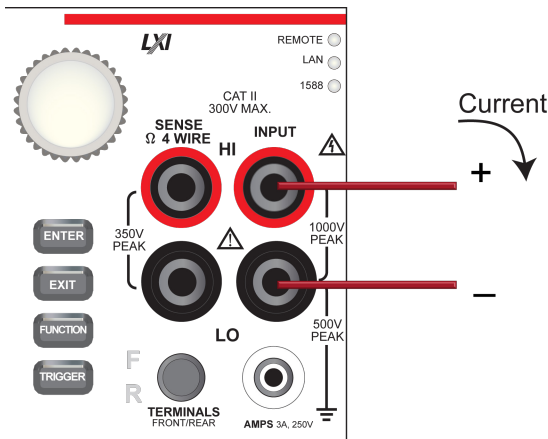
Figure 74: Rear panel connections: Continuity measurement
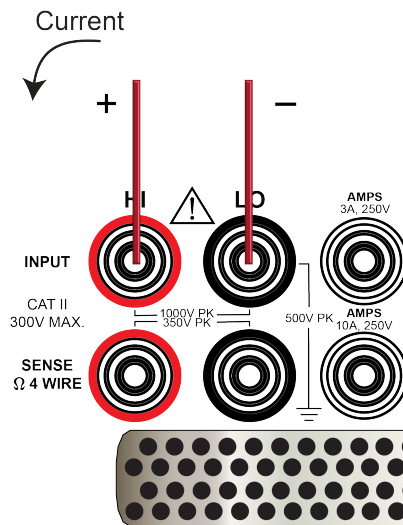
Measure continuity using the front panel

To make a continuity measurement using the front panel:

1.   Make the connections as shown in the previous figures.
2.   Press the **FUNCTION** key.
3.   Select **Continuity**.
4.   Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for continuity measurements

See Continuity measure settings (on page 2-29) for settings that are available when you are making continuity measurements.

Frequency measurements

This section describes how you can set up frequency measurements from the front panel. Frequency measurements are only applicable to voltage signals.

Frequency and period support fixed and autorange threshold ranging, with a range of 100 mV to 700 V. Ranges are scaled to RMS sine wave voltages.

When autorange is selected, there are two measurement phases, measure AC voltage and measure frequency or period. When the AC voltage is measured, the amplitude is measured and the appropriate range is selected to ensure 11 % to 110 % signal scaling. In the second phase, the frequency or period is measured.

Frequency and period are specified for square wave inputs. The input signal must be more than 1 0% of the AC voltage range. If the input is less than 20 mV and measured on the 100 mV range, the frequency must be more than 10 Hz. For sine wave inputs, the input frequency must be more than 100 Hz.

You can set a zero crossing threshold trigger level. The level can be set from -700 V to 700 V. The input signal is AC coupled to the zero crossing threshold level detection.

Threshold levels are scaled to 100 % peak of the selected range. For example, if the 1 V range is selected and the threshold level is set to 0.5 V, the frequency or period counter counts for portions of the input at more than 0.5 V, or 0.353 V_{rms} for a sine wave.

To use the threshold level, you must set the threshold range to a set value (it cannot be set to autorange).

⚠ CAUTION

Do not apply more than 1000 V_{peak} between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

Frequency measure connections

Figure 75: Front panel connections: Frequency measurement
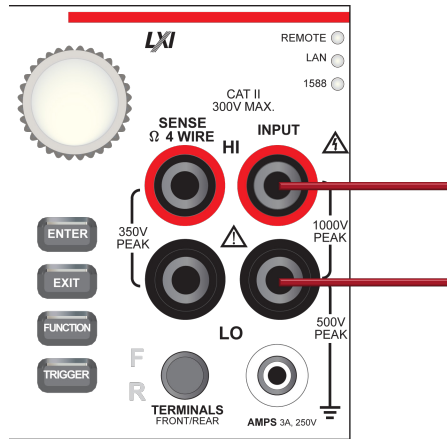
Figure 76: Rear panel connections: Frequency measurement
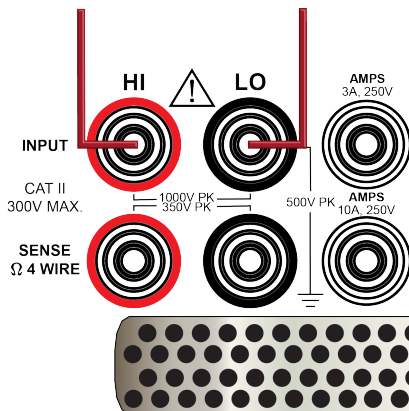
Measure frequency using the front panel

To make a frequency measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **Frequency**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for frequency measurements

See for settings that are available when you are making frequency measurements.

Period measurements

This section describes how you can set up period measurements from the front panel.

Period measurements are only applicable to voltage signals.

> ⚠ **CAUTION**
>
> Do not apply more than 1000 V$_{peak}$ between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

Period measure connections

Figure 77: Front panel connections: Period measurement
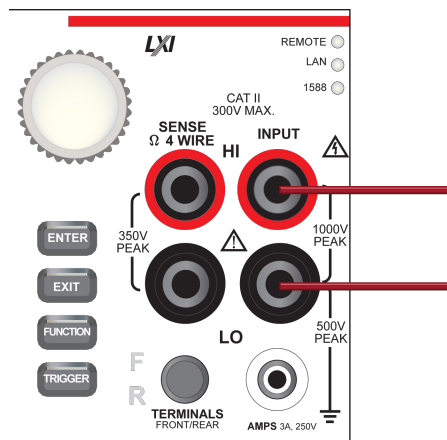
Figure 78: Rear panel connections: Period measurement
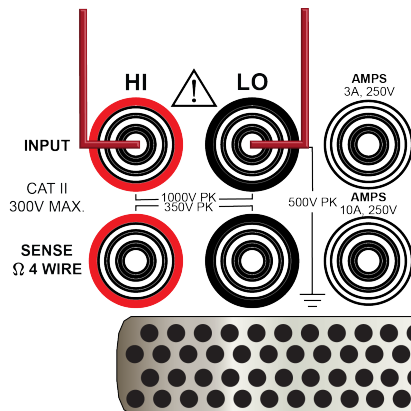
Measure the period using the front panel

To make a period measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **Period**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for period measurements

See Period measure settings (on page 2-30) for settings that are available when you are making Period measurements.

Diode measurements

With a Model DMM7510, you can measure the forward voltage drop of general-purpose diodes and the zener voltage of zener diodes. You can measure the forward voltage drop of a diode on the 10 V range with a constant test current (bias level). You can select a bias level of 10 μA, 100 μA, or 1 mA.

♀ Quick Tip

The diode function I_{test} current is very stable, but it is ±5 % actual. For simple I-V semiconductor applications, you can determine the actual I_{test} source value by using a remote command to return an accurate current for the bias level setting. If you are using the SCPI command set, refer to [:SENSe[1]]:<function>:BIAS:ACTual? (on page **6-69**). If you are using the TSP command set, refer to dmm.measure.bias.actual (on page **8-131**).

⚠ CAUTION

Do not apply more than 1000 V_{peak} between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

Diode measure connections

Figure 79: Front panel connections: Diode measurement
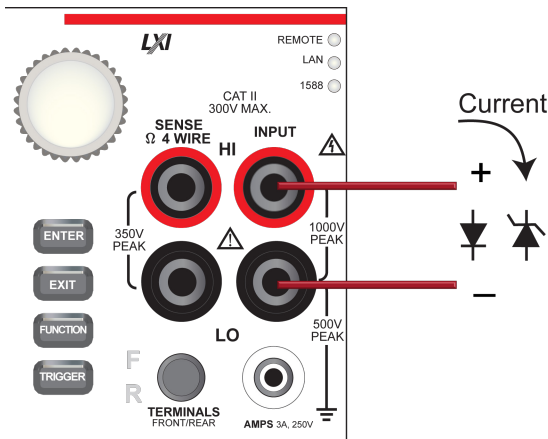
Figure 80: Rear panel connections: Diode measurement
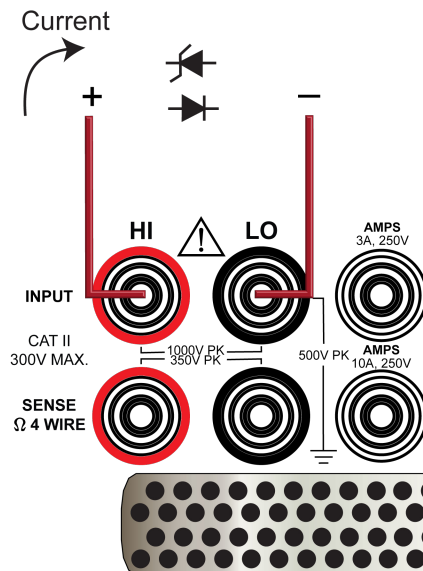
Measure diode forward bias using the front panel

To make a diode measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **Diode**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for diode measurements

See Diode measure settings (on page 2-30) for settings that are available when you are making diode measurements.

Temperature measurements

This section describes how to set up temperature measurements. You can measure temperature using thermocouples, thermistors, and 3-wire or 4-wire resistance temperature detectors (RTDs).

Temperature measure connections

Figure 81: 2-wire thermistor connections
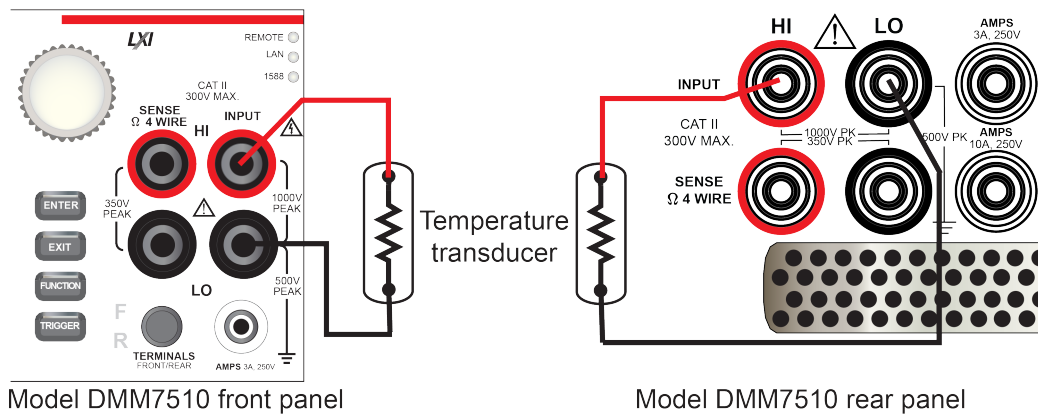
Model DMM7510 front panel   Model DMM7510 rear panel

Figure 82: 4-wire RTD measurement
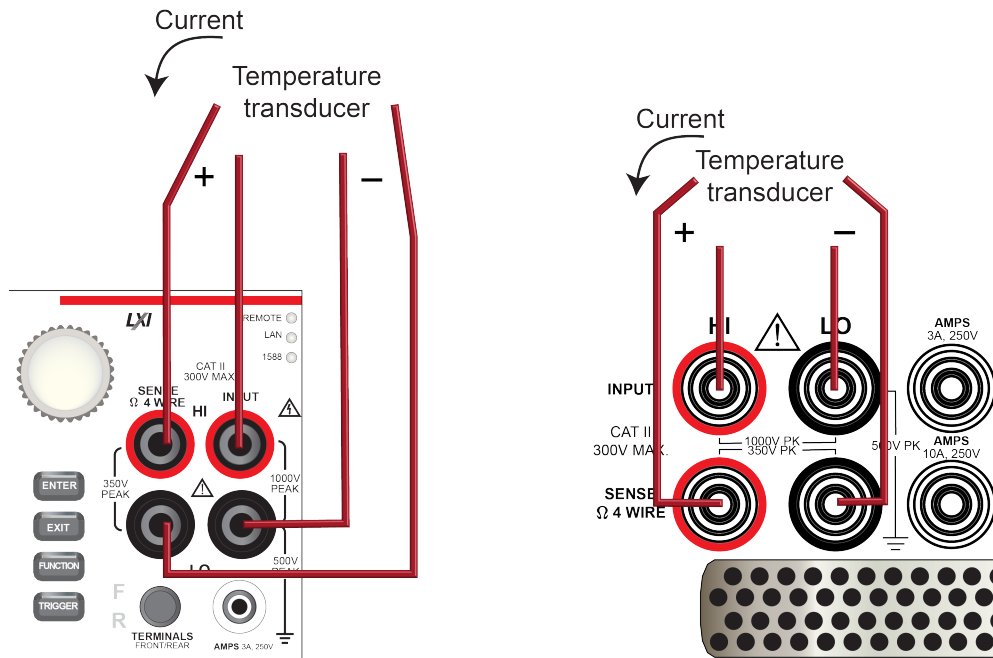
Model DMM7510 front panel   Model DMM7510 rear panel

Measure temperature using the front panel

To make a temperature measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **Temperature**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Temperature transducer types

You can use thermocouples, thermistors, 3-wire RTDs, and 4-wire RTDs with the Model DMM7510.

For thermocouples, temperature measurement range depends on which type of thermocouple is being used. Thermocouple B, E, J, K, N, R, S, and T are supported.

The thermistor types 2252, 5000, and 10,000 are supported. Note that curve-fitting constants are used in the equation to calculate thermistor temperature. The thermistor manufacturer's specified curve fitting may not be the same as the ones used by the Model DMM7510.

The Model DMM7510 supports 3-wire and 4-wire RTD types of:

- PT100
- D100
- F100
- PT385
- PT3916

You can also select the user type. When the user type is selected, you can define the alpha, beta, delta, and zero values of the RTD.

For 3-wire RTD measurements, the HI, LO, and SENSE LO input terminals are used to measure temperature. The SENSE LO remote senses lead resistance and properly compensates the resistance measurement before converting to temperature. The accuracy for 3-wire RTD is with less than a 0.1 Ω lead resistance mismatch for INPUT HI and INPUT LO. Add 0.25 °C per 0.1 Ω of HI-LO lead resistance mismatch.

For 4-wire RTD measurements, by default, the Model DMM7510 measures temperature with offset-compensated ohms and open lead detection enabled. This provides the most accurate and reliable method to measure the low resistance of the RTD. For faster RTD measurements when the most accurate measurements are not required, you can disable offset compensation and open lead detection for 3-wire and 4-wire RTD measurements.

Settings available for temperature measurements

See Temperature measure settings (on page 2-31) for settings that are available when you are making temperature measurements.

Capacitance measurements

With a Model DMM7510, you can measure capacitance.

The capacitance function sources a constant I_{test} current through the device under test (DUT) while measuring voltage (dV) in a fixed time internal (dt). The Capacitance measurement is:

I_{test} * dt / dV

Capacitance measurements have two measurement phases, discharge and charge. During the discharge phase, the DUT is connected through an internal 13 mA current source and discharged to approximately 0 V. In the charge phase, the I_{test} is sourced while measuring the voltage. If the voltage on the DUT exceeds 2.8 V ±10 %, the I_{test} is halted and the voltage is held until the discharge phase. If the voltage is less than 2.8 V, the resultant capacitance measurement is calculated.

Capacitance supports 1 nF to 1 mF ranges. Each range measures from 0 % to 120 % full scale. Reading rates vary based on range and the percent of full scale.

The 13 mA discharge and I_{test} currents are protected to 1000 V.

Capacitance has a fixed aperture time.

> ⚠ CAUTION
>
> Do not apply more than 1000 V_{peak} between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

Capacitance measure connections

Figure 83: Front panel connections: Capacitor measurement
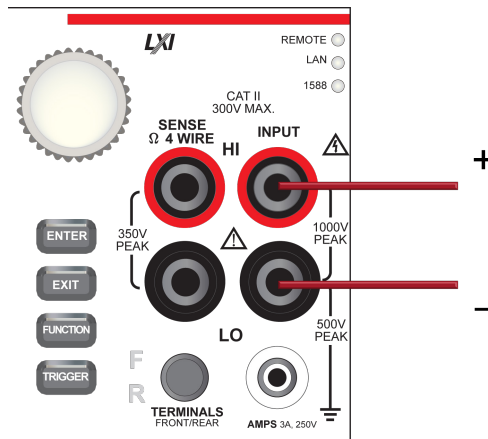
Figure 84: Rear panel connections: Capacitor measurement
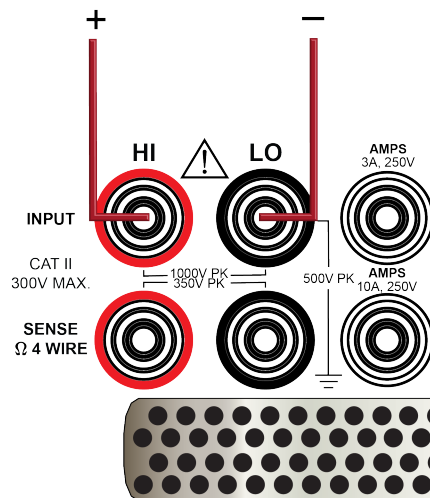
Measure capacitance using the front panel

To make a capacitance measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **Capacitance**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for capacitance measurements

See <u>Capacitance measure settings</u> (on page 2-32) for settings that are available when you are making capacitance measurements.

DC voltage ratio measurements

The DC voltage ratio function calculates the ratio between the measure input (numerator) and the reference voltage (denominator). This function can be useful when comparing one or more voltages to a single voltage. Only DC voltages can be compared.

The SENSE terminals are used as the reference voltage (V_s). The SENSE terminals can measure DC volts in 100 mV, 1 V, and 10 V ranges.

The INPUT terminals provide the voltage (V_i) to be compared against the reference voltage. They can measure DC volts in 100 mV, 1 V, 10 V, 100 V, and 1000 V ranges.

The ratio is calculated as:

$$\text{Ratio} = \frac{V_{input} - V_{input\_rel}}{V_{sense} - V_{sense\_rel}}$$

⚠ CAUTION

SENSE HI and LO must be referenced to INPUT LO.

SENSE HI must not exceed 125 %, referenced to INPUT LO, of the selected sense range.

NOTE

To access the extra value in the reading buffer, the reading buffer style must be set to full. The extra value is available through the front panel in the Reading Details, through the SCPI command :TRACe:DATA? (on page 6-147), and through the TSP command bufferVar.extravalues (on page 8-28). Refer to Creating buffers (on page 3-14) for information on setting the reading buffer style to full.

For example, if you have a 9 kΩ/1 kΩ resistive network, connect a 1 V source across the network. Connect measurement Input HI and LO across the total 9 kΩ/1 kΩ resistive network and select the 1 V measure range. Connect Sense HI and LO across the 1 kΩ portion of the network and select the 100 mV range. The ratio measurement is approximately 10.00000.

Figure 85: DCV ratio 9 kohm/1 kohm resistor network example
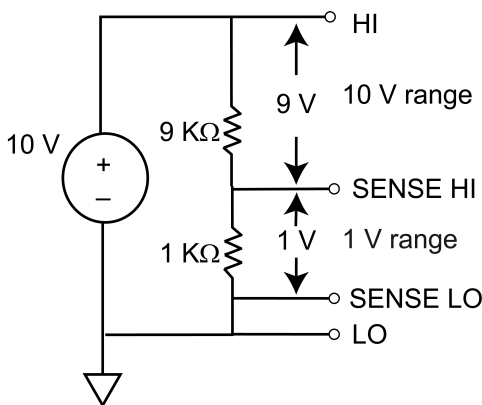
Another example is a 1 kΩ/1 kΩ/1 kΩ resistor network. If 3 V is applied across the total three 1 kΩ resistors and V_{sense} is across the first 1 kΩ resistor, set V_{input} to the 10 V range and V_{sense} to the 1 V range. The ratio measurement is approximately 3.00000. If V_{sense} is set to the 1 V range, the ratio displays overflow, with SENSE HI and SENSE LO terminals exceeding the 125 % maximum reference to the LO terminals. The SENSE HI to LO is 3 V and SENSE LO to LO is 2 V, respectively.

Figure 86: DCV ratio 1 kohm/1 kohm/1 kohm resistor network
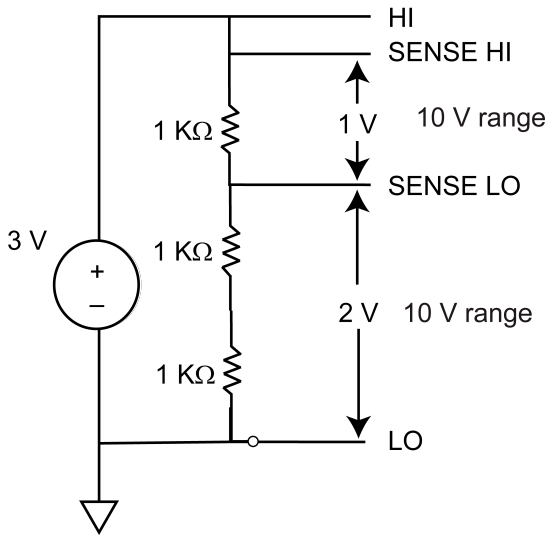
> ⚠ **CAUTION**
>
> Do not apply more than 1000 V_{peak} to the INPUT terminals or more than 350 V_{peak} to the SENSE terminals. Failure to heed this caution may result in instrument damage.

DC voltage ratio measure connections

Figure 87: Front panel connections: DC voltage ratio measurement
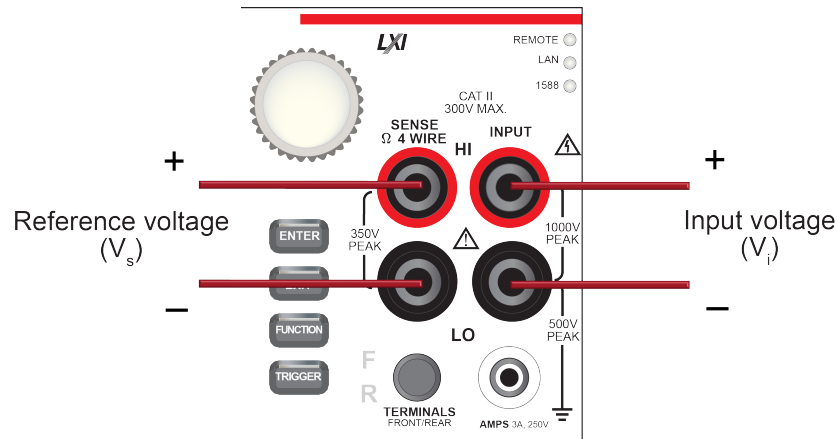
Figure 88: Rear panel connections: DC voltage ratio measurement
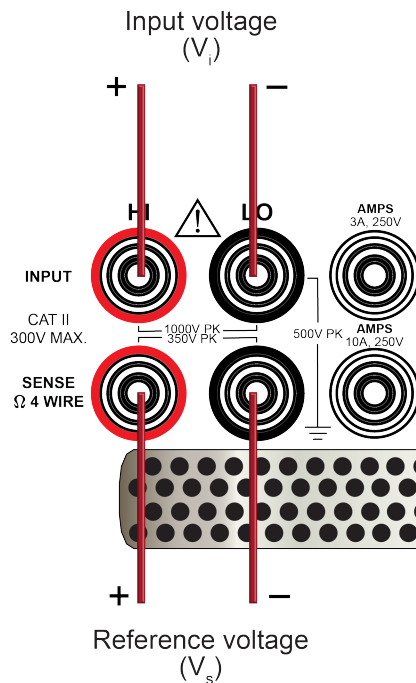
Measure DC voltage ratio using the front panel

To make a DC voltage ratio measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select **DCV Ratio**.
4. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for DC voltage ratio measurements

See DC voltage ratio measure settings (on page 2-32) for settings that are available when you are making continuity measurements.

Digitize functions

The Model DMM7510 digitize functions make fast, predictably spaced measurements. The speed, sensitivity, and bandwidth of the digitize functions allows you to make accurate voltage and current readings of fast signals, such as those associated with sensors, audio, medical devices, power line issues, and industrial processes. The digitize functions can provide 1,000,000 readings per second at 4½ digits. Digitize voltage and digitize current have separate internal signal paths that are optimized for fast response to signal changes.

The sample rate determines how often the readings are output by the digitize function. You can set it from 1000 to 1,000,000 readings per second.

The aperture determines the reading conversion time. This is when data is gathered to create the reading. You set the aperture time in 1 μs intervals. If the aperture is more than 1 μs, the observed data is averaged to produce the reading.

The sample rate affects the available aperture settings. The maximum aperture is determined by 1/sample rate (rounded down to the nearest integer). The instrument will automatically adjust the aperture settings if the sample rate is changed to a rate that does not support the existing aperture setting. When this occurs, a warning message is generated that reports the new aperture setting.

The count is the number of times to make readings with the selected sample rate and aperture after a trigger is detected. In continuous mode, the instrument generates automatic triggers. In manual mode, a trigger is defined by pushing the TRIGGER key on the front panel. You can also set up other types of triggers. For more information on triggers, refer to Triggering (on page 3-59).

If you are using the TSP command language, the commands use a different syntax for measure and digitize commands. For example, command to change the measure function range is:

```
dmm.measure.range = 100
```

The command to change the digitize range is:

```
dmm.digitize.range = 100
```

Digitize voltage and digitize current support the same ranges as DC voltage (100 mV to 1000 V) and DC current (10 μA to 3 A with front panel connections; 10 μA to 10 A with rear panel connections). Digitize function do not support autorange, autozero, or auto delay.

Digitize voltage measurements

The digitize voltage function makes accurate, predictably spaced voltage measurements.

⚠ **CAUTION**

Do not apply more than 1000 V$_{peak}$ between INPUT HI and LO. Failure to observe this caution may result in instrument damage.

Digitize voltage measure connections

The connections for front-panel and rear-panel digitize voltage measurements are shown in the following graphics.

Figure 89: Front panel connections: Digitize voltage measurement
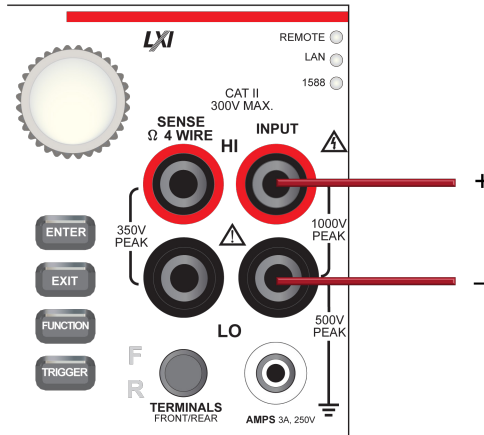
Figure 90: Rear panel connections: Digitize voltage measurement
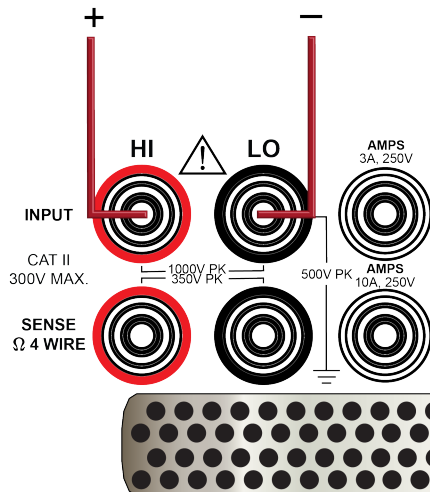
Measure with digitize voltage using the front panel

To make a digitize voltage measurement using the front panel:

1. Make the connections as shown in Digitize voltage measure connections (on page 2-125).
2. Press the **FUNCTION** key.
3. Select the **Digitize Functions** tab.
4. Select **Digitize Voltage**.
5. Press the **MENU** key.
6. Under Measure, select **Settings**.
7. Select the settings for your application. For descriptions of the options, refer to Digitize Voltage measure settings (on page 2-33).
8. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements display on the front panel.

Settings available for digitize voltage measurements

See Digitize voltage measure settings (on page 2-33) for settings that are available when you are digitizing voltage measurements.

Digitize current measurements

The digitize current function makes accurate, predictably spaced current measurements.

⚠ CAUTION
Do not apply more than 250 V_{peak} between INPUT LO and the AMPS input. Failure to observe this caution may result in instrument damage.

Digitize current measure connections

Figure 91: Front panel connections: Digitize current measurement
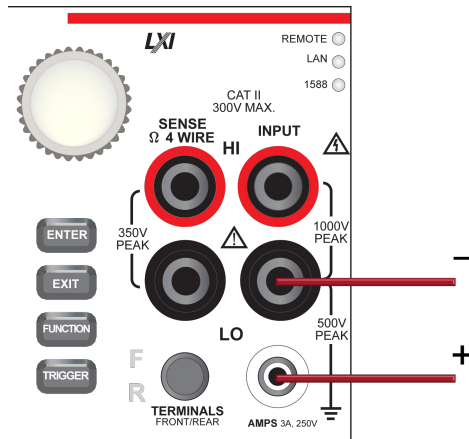
Figure 92: Rear panel connections: Digitize current measurement (current below 3 A)
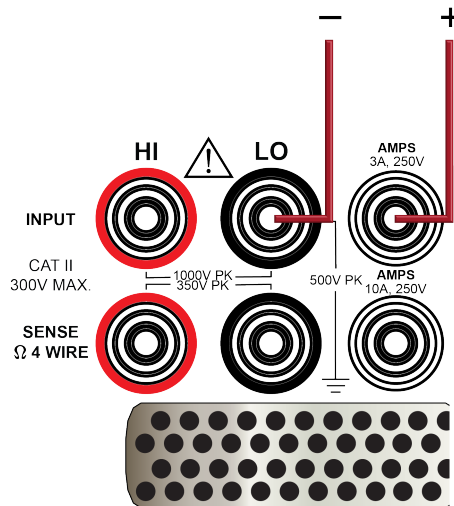
Figure 93: Rear panel connections: Digitize current measurement (current below 10 A)
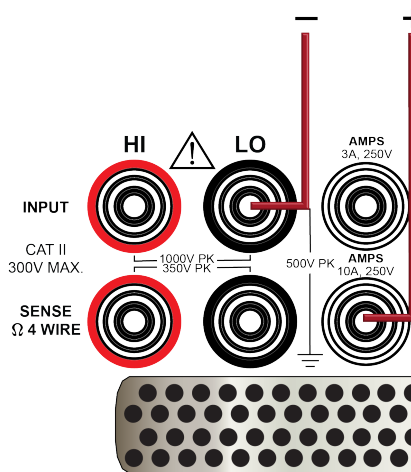
Measure with digitize current using the front panel

To make a digitize current measurement using the front panel:

1. Make the connections as shown in the previous figures.
2. Press the **FUNCTION** key.
3. Select the **Digitize Functions** tab.
4. Select **Digitize Current**.
5. Press the **TRIGGER** key for 2 seconds and verify that the instrument is set to Continuous Measurement.

The measurements start displaying on the front panel.

Settings available for digitize current measurements

See Digitize current measure settings (on page 2-33) for settings that are available when you are digitizing current measurements.

Digitizing aperture and sample rate

In most cases, you will get good results if you leave the aperture at the default setting of automatic. When auto is selected, the instrument makes as many measurements as possible in the sample period. When it is set automatically, it is set to 1 mega sample/second (rounded down to the nearest integer).

You might want to set a manual aperture if you need an aperture that contains a higher than one discrete 1 μs averaged reading.

Although the maximum sample rate is 1 million samples per second, the input filtering of the A/D is set at a 3 dB corner point of slightly greater than 350 kHz to prevent aliasing. Therefore, a 350 kHz or higher voltage input is attenuated by a factor of 0.707. For dynamic signals, this attenuation could cause attenuated readings. Consult the specifications for detail.

Input frequencies above 500 kHz are occasionally prone to the signal processing problem of aliasing.

The following figure shows the relationship between the aperture, sample rate, and count.

Figure 94: Digitize aperture, sample rate, and count
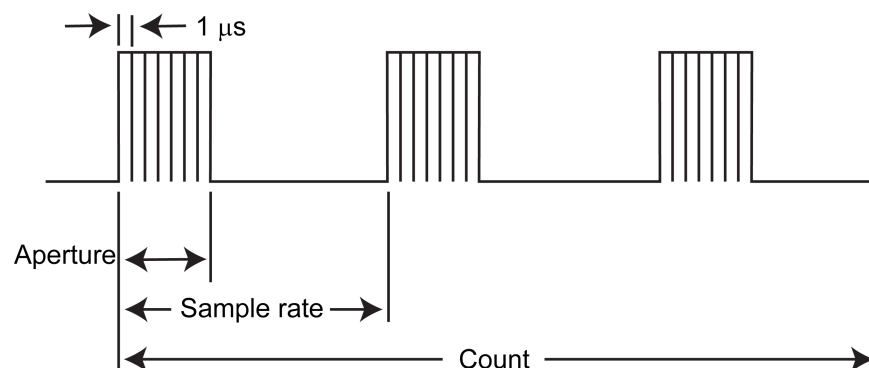
For large count (more than 8,000,000) and sample rate values (more than 150,000), data may lost. Adjust one of the values to a lower level.

DC and AC coupling

When the function is set to digitize voltage, you can set the signal coupling mode to be DC or AC.

When DC coupling is selected, the instrument measures all AC and DC content of the input signal. Measure resolution is 1 μV to 1000 V.

If input impedance is set to automatic, DC coupling provides the lowest measurement noise and highest isolation to measure loading, more than 10 GΩ for the 100 mV to 10 V ranges. The 100 V and 1000 V ranges terminate a 10 MΩ divider across HI and LO input terminals.

If input impedance is set to 10 MΩ, the 100 mV to 1000 V ranges are terminated with a 10 MΩ divider across the HI and LO input terminals. This provides stable readings with open input terminals (approximately 1 mV). The 100 V and 1000 V ranges have the best bandwidth (approximately 20 kHz). The 100 mV to 10 V ranges have consistent 600 kHz bandwidth with either input impedance setting.

When AC coupling is selected, the instrument only measures the AC content of the input signal. For AC coupling, the Model DMM7510 is optimized for all ranges with a consistent 600 kHz bandwidth.

AC coupling is terminated with an RC filter that can be set to fast or slow. Internally, R is fixed at 1.1 MΩ across the HI and LO terminals. This optimizes bandwidth and measure loading.

When the fast filter is selected, the Model DMM7510 is optimized to settle in 85 ms with AC waveforms that contain less than or equal to 3 % DC content. You can achieve additional DC content up to 400 V, but additional settling time may be needed to achieve rated accuracies.

Fast AC coupling is recommended for signals that are more than or equal to 1 kHz.

When the slow filter is selected, the Model DMM7510 is optimized to settle in 850 ms with AC waveforms that contain less than or equal to 3 % DC content. You can achieve additional DC content up to 400 V, but additional settling time may be needed to achieve rated accuracies.

Slow AC coupling is recommended for signals that are less than or equal to 3 Hz to 600 kHz.

You can properly amplitude-compensate the digitize voltage AC coupling readings by entering the frequency of the input signal. For example, to digitize a 2 V peak-to-peak signal at 3 Hz, set the AC coupling filter to slow and enter 3 Hz for the AC coupling frequency. This properly compensates the attenuation from a low frequency signal across the internal RC filter.

You can compensate the reading for both slow and fast AC coupling filters. The allowed frequencies are from 3 MHz to 1 MHz.

Display results of two measure functions

The Model DMM7510 allows you to make and display two measurements from different functions. The measurements are displayed on the front panel and stored in the reading buffers.

To access the dual measurement capability, swipe the lower half of the home screen to the SECONDARY swipe screen. This feature is only available from the front panel of the instrument.

NOTE
Depending on the selected functions, a relay may click when the instrument switches between the measurement types. Leaving secondary measurements on for extended periods may shorten the life of the relays.

Figure 95: SECONDARY swipe screen
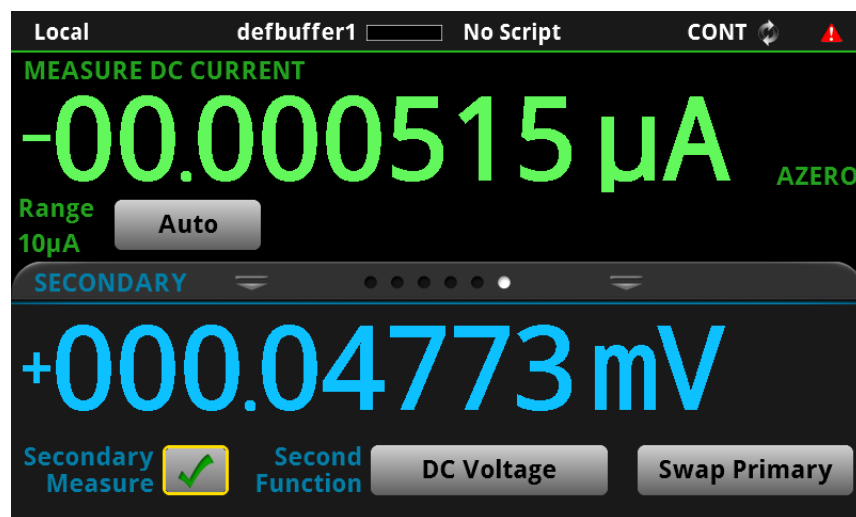
Making secondary measurements

When you are using the secondary measurements feature, any settings that you change from the front panel of the instrument affect the primary function (the function shown at the top of the Secondary swipe screen). To change settings for the secondary function (the function shown at the bottom of the Secondary swipe screen), you need to swap the functions. Select the **Swap Primary** button to switch the primary and secondary functions. Changes made for a particular measure function while in the primary will remain set for that measure function until specifically changed.

Measurements are stored in separate reading buffers. By default, the primary measurements are stored in defbuffer1 and secondary measurements are stored in defbuffer2. For the primary measurement, you can change the reading buffer by making it active. Refer to Using the front panel to select a reading buffer (on page 3-17) for detail. You cannot change the reading buffer for the secondary measurement.

Secondary measurements are not available for use with the trigger model.

To make secondary measurements:

1. Make connections to the instrument appropriate to both types of measurements. Refer to DMM measurement overview (on page 2-91) for connection information.
2. Swipe to the SECONDARY swipe screen.
3. Set up the primary function as needed.
4. Select Second Function to select the secondary function.
5. If you need to change settings for the secondary function, select **Swap Primary**. Make the settings as needed, then select **Swap Primary** again.
6. Hold the **TRIGGER** key for 2 seconds and select **Continuous Measurement** or **Manual Trigger Mode**.
7. Select **Secondary Measure**.
8. If you selected Continuous Measurements, measurements for both functions begin. If you selected Manual Trigger Mode, measurements are made when you press the **TRIGGER** key.

Displayed measurements

When you make measurements, the instrument may perform operations on the measured values that affect what you see on the display and the measurements that are stored in the buffer.

The operations that can affect the measurement display are:

- Filtering
- Relative offset
- Math operations
- Limit tests

If none of these operations is set, the value that is displayed on the front panel is the actual measurement reading.

If any one of these operations is set, the value that is displayed is the measurement reading with these operations applied. The operations are applied in the order shown above.

For example, if you made a measurement and had a relative offset and limit tests active, the measured value would have the relative offset applied, then have limit test results applied.

For additional detail on the order of operations, see Order of operations (on page 4-16).

Using Quick Setups

The Quick Setup menu includes options that allow you to change the measure function, adjust the performance, and select Quick Setups.

The measure functions available through the QuickSet menu include the same functions that are available through the front-panel **FUNCTION** key.

Using the Performance slider

Use the Performance slider to adjust for performance (resolution versus speed).

When you adjust the Performance slider, the instrument changes settings based on where you position the slider. As you increase reading speed, you lower the amount of resolution. As you increase resolution, you decrease the speed. These settings take effect the next time measurements are made.

Note that if the instrument is set to the DC voltage, DC current, digitize voltage, or digitize current function, changing the speed may change the function from the DC voltage or DC current to the digitize voltage or digitize current function and vice versa.

When the temperature function is selected, the readings per second are shown as a range to accommodate the various transducer types.

Making a measurement with the QuickSet functions

To use a Quick Setup, make connections, then press **QUICKSET** and select a Quick Setup. You are prompted for the settings for that Quick Setup.

The Quick Setups include Voltage Waveform, Current Waveform, Interval Measure, and External Scan. Descriptions of the Quick Setups and brief information about the settings are provided in the following topics.

Voltage Waveform Quick Setup

The **Voltage Waveform** Quick Setup helps you set up Digitize Voltage measurements and displays the results on the Graph screen. You are prompted to set the sample rate, sample count, and signal amplitude when you select the Voltage Waveform Quick Setup. The instrument is reset before these settings are applied.

The sample rate defines the precise acquisition rate at which the digitizing measurements are made.

The count sets the number of measurements to digitize when a measurement is requested.

The signal amplitude allows the instrument to select a fixed range that is large enough to measure your signal.

When the settings are complete, the Graph screen is displayed with the measure results.

Current Waveform Quick Setup

The Current Waveform Quick Setup helps you set up Digitize Current measurements and displays the results on the Graph screen.

You are prompted to set the sample rate, count, and signal amplitude when you select the Current Waveform Quick Setup. The instrument is reset before these settings are applied. The instrument also verifies whether the front or rear terminals are selected. If the front terminals are selected, signal amplitude is limited to 3 A; if the rear terminal is selected, amplitude is limited to 10 A.

The sample rate defines the precise acquisition rate at which the digitizing measurements are made.

The count sets the number of measurements to digitize when a measurement is requested.

The signal amplitude allows the instrument to select a fixed range that is large enough to measure your signal.

When the settings are complete, the Graph screen is displayed with the measure results.

Interval Measure Quick Setup

The Interval Measure Quick Setup helps you set up measurements that occur at precisely timed intervals. The settings you make depend on whether you are using a measure function or a digitize function.

When you run this Quick Setup for a measure function, the instrument:

- Is reset
- Prompts you to select a function
- Prompts you for the interval time
- Prompts you for the number of readings (count)
- Sets up the trigger timer and trigger model
- Initiates the trigger model
- Displays measurements on the home screen and stores them in `defbuffer1`

When you run this Quick Setup for a digitize function, the instrument:

- Is reset
- Prompts you for the interval (sample rate)
- Prompts you for the number of readings (count)
- Sets up a trigger model
- Initiates the trigger model
- Displays the measurements on the home screen and stores them in `defbuffer1`

External Scan Quick Setup

The External Scan Quick Setup helps you set up the instrument to communicate with an external switch system.

This Quick Setup does not reset the instrument, so before running it, make sure all function settings are configured as needed for the measurements that will be made.

When you run this Quick Setup, the instrument:

* Clears the reading buffer

* Prompts you for the type of in and out lines (digital I/O or external I/O)

* Prompts you for the number of channels in a scan

* Prompts you for the number readings per channel

* Prompts you for the function that will be run for each channel

Using this information, the instrument sets up configuration lists and the I/O lines for use with the switch system.

A trigger out from the instrument notifies the switch system to control the scan channels.

A trigger in from the switch system indicates that the switch is ready and the instrument can make measurements.

Auto Delay

Auto Delay applies a wait period at the end of a function change, range change, and other measure-related settings. The delay allows cables or internal DMM circuitry to settle for best measurement accuracy. For the AC current and AC volts functions, the autodelay includes both the RMS filter and AC-coupling capacitor-settling times.

* When autodelay is disabled, no wait time is applied.

* When autodelay is enabled, a measurement is not made until immediately after the autodelay period has expired. Depending on the length of the required delay, there may or may not be an impact on the first reading after a function or range change.

Autodelay times for each function are provided in the following topics.

To set autodelay from the front panel for the selected function:
1. Press the **MENU** key.
2. Select Measure **Settings**.
3. Next to Auto Delay, select **On** to include a delay or **Off** to remove the delay.

To set autodelay using SCPI commands:

Refer to [:SENSe[1]]:<function>:DELay:AUTO (on page 6-76)

To set autodelay using TSP commands:

Refer to dmm.measure.autodelay (on page 8-127)

Voltage autodelay and autorange times

The following table provides times for autodelay and autorange for the Model DMM7510 DMM voltage functions.

Function	Detector bandwidth	Range and delays					
DC volts	Not applicable	Range	100 mV	1 V	10 V	100 V	1000 V
		Autodelay	1 ms	1 ms	1 ms	5 ms	5 ms
		Autorange	10 ms	10 ms	10 ms	50 ms	50 ms
AC volts	Not applicable	Range	100 mV	1 V	10 V	100 V	700 V
	3 or 30 Hz	Autodelay	265 ms	265 ms	265 ms	265 ms	1 s
		Autorange	2.65 s	2.65 s	7.5 s	7.5 s	10 s
	300 Hz	Autodelay	50 ms	50 ms	50 ms	50 ms	250 ms
		Autorange	3.5 ms	3.5 s	850 ms	850 ms	2.5 s

Current autodelay and autorange times

The following tables provide times for autodelay and autorange for the Model DMM7510 DMM current functions.

Function	Range and delays								
DC current	Range	10 μA	100 μA	1 mA	10 mA	100 mA	1 A	3 A	10 A
	Autodelay	13 ms	2 ms	2 ms	2 ms	2 ms	2 ms	2 ms	2 ms
	Autorange	130 ms	20 ms	20 ms	20 ms	20 ms	20 ms	20 ms	20 ms

Function	Detector bandwidth	Range and delays						
AC current	Not applicable	Range	1 mA	10 mA	100 mA	1 A	3 A	10 A
	3 or 30 Hz	Autodelay	265 ms	265 ms	265 ms	265 ms	300 ms	300 ms
		Autorange	2.65 s	2.65 s	2.65 s	2.65 s	3.0 s	3.0 s
	300 Hz	Autodelay	50 ms	50 ms	50 ms	50 ms	75 ms	75 ms
		Autorange	500 ms	500 ms	500 ms	500 ms	750 ms	750 ms

Resistance autodelay and autorange times

The following tables provide times for autodelay and autorange for the Model DMM7510 DMM resistance functions.

For continuity, the range is 1 kΩ with an autodelay of 3 ms and auto range of 25 ms.

Function	Range and delays								
2-wire ohm and 4-wire ohm	**Range**	1 - 100 Ω	1 kΩ	10 kΩ	100 kΩ	1 MΩ	10 MΩ	100 MΩ	1 GΩ
	Autodelay	3 ms	3 ms	13 ms	25 ms	100 ms	250 ms	375 ms	375 ms
	Autorange	2.5 ms	25 ms	125 ms	250 ms	1 s	2.5 s	3.75 s	3.75 s

Function	Range and delays		
Dry circuit ohms	**Range**	1 - 10 Ω	100 - 2 kΩ
	Autodelay	3 ms	13 ms
	Autorange	2.5 ms	125 ms

Frequency and period autodelay and autorange times

The following table provides times for autodelay and autorange for the Model DMM7510 DMM frequency and period functions.

Function	Ranges and delays					
Frequency and periods	**Range**	100 mV	1 V	10 V	100 V	700 V
	Autodelay	100 ms				
	Autorange	1 s				

Temperature autodelay and autorange times

The following table provides times for autodelay and autorange for the Model DMM7510 DMM thermistor and RTD temperature functions.

When thermocouple is selected, the range is 100 mV, with an autodelay of 1 ms and an autorange of 10 ms.

Function	Range and delays				
Thermistor	Range	1 Ω – 100 Ω	1 kΩ	10 kΩ	100 kΩ
	Autodelay	3 ms	3 ms	13 ms	25 ms
	Autorange	25 ms	25 ms	125 ms	250 ms
3-Wire and 4-Wire RTD	Range	100 Ω	1 kΩ	10 kΩ	100 kΩ
	Autodelay	3 ms	3 ms	13 ms	25 ms
	Autorange	25 ms	25 ms	125 ms	250 ms

Capacitance autodelay and autorange times

The following table provides times for autodelay and autorange for the Model DMM7510 DMM capacitance functions.

Function	Range and delays						
Capacitance	Range	1 nF	10 nF	100 nF	1 μF	100 μF	1 mF
	Autodelay	1 ms					
	Autorange	10 ms					

Diode autodelay and autorange times

The following table provides times for autodelay and autorange for the Model DMM7510 DMM diode functions.

Function	Range and delays		
Diode	Bias level	10 μA	100 μA	1 mA
	Autodelay	1 ms	1 ms	1 ms
	Autorange	10 ms	10 ms	10 ms

DCV ratio autodelay and autorange times

The settle time for the DCV ratio function is the same as the settle time for the larger of the two voltage ranges.

Detector bandwidth

You can select the detector bandwidth AC volt and AC current measurements. You can select 3 Hz, 30 Hz, or 300 Hz.

When you select the 3 Hz bandwidth, the signal goes through an analog root-mean-square (RMS) converter. The output of the RMS converter goes to a fast (1 kHz) sampling A/D and the RMS value is calculated from 1200 digitized samples (1.2 s).

When you select the 30 Hz bandwidth is chosen, the same converter is used. However, only 120 samples (120 ms) are needed for an accurate calculation because the analog RMS converter has turned most of the signal to DC.

When you select the 300 Hz bandwidth, the output of the analog RMS converter (nearly pure DC at these frequencies) is measured at an integration rate of 16.6 ms (1 power line cycle). You can set the integration rate from 8.333 μs to 0.25 ms (60 Hz) and 10 μs to 0.24 ms (50 Hz).

To achieve the best accuracy for AC volt and AC current measurements, use the bandwidth setting that best reflects the frequency of the input signal. For example, if the input signal is 40 Hz, a bandwidth setting of 30 should be used.

NOTE

You can only adjust the NPLC or aperture for AC voltage and AC current measurements when the bandwidth for that function is set to 300 Hz.

Graphing

The graphing features of the Model DMM7510 allow you to view your measurement data graphically. You can compare up to four traces on the front panel of the instrument. You can manipulate the graph to view minimums and maximums, view averages, determine deltas, and view the values of specific data points.

Setting up the Graph tab

When you start up the instrument, the Graph tab plots data from the active reading buffer as measurements are made. You can change which data is displayed, how it is scaled, and what kinds of triggers are used to generate measurements. These settings are changed on the Data, Scale, and Trigger tabs.

Selecting the data to be plotted

The graph plots data from reading buffers. When you first open the Graph screen, the data from the active reading buffer is plotted. You can change the buffer to display different data, or select multiple buffers to display multiple traces on the graph. Each trace represents the data from one reading buffer.

To select the buffer that contains the data:

1. Press the **MENU** key.
2. In the View menu, select **Graph**.
3. Select the **Data** tab. The Trace Data table displays the buffers that are presently plotted on the Graph tab.
4. To add an additional buffer, select **Add Trace** and select the reading buffer to add. You can select up to four buffers.
5. To remove a trace, select a buffer from the Trace Data table and select **Remove Trace**.
6. Set the marker for the data. The markers determine how data is represented when there are many data points. You can select:
   - Min/Max: The minimum to maximum points of the data are displayed as a solid vertical line.
   - Average: The data is averaged and displayed as a single averaged data point.
   - Both: The Min/Max line and the averaged data point are displayed.

NOTE

If you select the active buffer, the trace is set to be the reading buffer and the label "Active" is removed. The Graph tab will no longer switch to the new buffer if you change the active buffer. To return to plotting data from the active buffer, use Remove Trace to remove all of the traces. When the active buffer setting is on, "Active" is displayed before the name of the active buffer. When a trace is using the active buffer and the active buffer changes, the graph replots the data from the newly designated active buffer.

Trace colors

The colors for the traces are initially established in the order green, blue, orange, and brown. Once a color has been assigned to a trace, the color remains associated with that trace. For example, trace 3 is assigned to orange. If trace 1 and 2 are removed, trace 3 becomes trace 1, but remains orange on the Graph tab. New traces are assigned the next available color. In this example, if you add a trace, it is set to green.

Setting up scaling

The Scale tab allows you to set up how the data is displayed and tracked on the Graph tab.

The X-Axis Method and Y-Axis Method determine how the data is scaled and tracked on the Graph tab. The options are:

- SmartScale[TM]: The instrument scales the graph automatically. The instrument determines the best scale and tracking method based on the data, reading groups, number of traces, and instrument configuration. The scale is set to show the most relevant portion of the data that is in the selected reading buffer onto the screen.

- Track Latest: The graph always displays the latest data on a fixed scale.

- Track Group: The graph always displays shows the entire data group on the graph. The start of the group is indicated by a small triangle. A group is automatically created when the measure or digitize count is set to more than 1.

- All: All data in the buffer is displayed on the graph.

- Off: The graph is not automatically adjusted. You can adjust the data manually by swiping, pinching, and zooming. You can also set the Scale and Minimum Position on the Scale tab.

If multiple traces are selected, the Y-Axis Method also allows:

- Per Trace: Y-Axis only. Scales the Y-axis of the trace so it fits the entire height of the screen. Only available if multiple traces are selected. Traces may overlap.

- Lanes: Y-Axis only. Scales the Y-axes of the traces in equal, non-overlapping portions of the height of the screen. Only available if multiple traces are selected.

- Shared: Y-Axis only. Scales the Y-axis so that the minimum and the maximum are shared across all traces.

The X-Axis and Y-Axis scale allows you set the units per division for the axes.

The Minimum Position sets the first value that is visible on the graph for the selected trace. When you set a Minimum Position, the Method is set to off.

For the Y-Axis, you can set the scale format to linear or log. Select Linear to increase the step size in even increments. Select Log to increase the step size exponentially.

When multiple traces are selected, you can set the scale and minimum position of the Y-Axis for each trace. Select the **Trace** button to toggle between the traces and settings.

Setting up triggers

You can use the Trigger tab of the Graph menu to begin a single acquisition of data based on different trigger sources.

NOTE

This section describes in general how to set up triggering. It does not describe details on the trigger sources.

For detail on the digital I/O, refer to Digital I/O (on page 3-43).

For detail on the External I/O, refer to External I/O (on page 3-55).

For detail on TSP-Link, refer to TSP-Link System Expansion Interface (on page 3-102).

For detail on analog triggers, refer to Analog triggering overview (on page 3-60).

You can set triggers to be generated from the:

- Display TRIGGER Key: The trigger occurs when you press the TRIGGER key.
- External Digital: The trigger occurs when an external stimulus is detected. The external pulse can come from a digital input line, TSP-Link input line, or the rear-panel external input line.
- Waveform: Select an analog edge, pulse, or window to trigger. Analog triggers are only available for the DC voltage, DC current, digitize voltage, and digitize current functions.

When you set up triggers through the Trigger tab, the instrument defines the LoopUntilEvent trigger model template with the trigger settings. Readings are placed in the active reading buffer. If a trigger model exists, it is replaced by the new settings.

When you set up the Trigger tab, settings are not applied if you press the **EXIT** key. However, the settings are retained and displayed when you return to the Trigger tab. Select another Graph tab or press **MENU** or **HOME** key to leave the Trigger tab and apply the changes.

To set up triggers to occur when the front-panel TRIGGER key is pressed:

1. Press the **MENU** key.
2. In the View menu, select **Graph**.
3. Select the **Trigger** tab.
4. Set the Source Event to **Display TRIGGER key**.
5. Set the length of the **Delay** that occurs before each measurement. 0 sets no delay.
6. Set the **Position**. The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled.
7. To start the measurements, press the **TRIGGER** key.
8. Select the **Graph** tab to view the readings.
9. Press the **TRIGGER** key to initiate a trigger. The trigger point is indicated by a small triangle above the point.

To set up triggers to occur based on input trigger pulses:

1. Press the **MENU** key.
2. In the View menu, select **Graph**.
3. Select the **Trigger** tab.
4. Set the Source Event to **External**.
5. Select **Digital Input**, **TSP-Link Input**, or **External Input**.
6. Set the length of the **Delay** that occurs before each measurement. 0 sets no delay.
7. Set the **Position**. The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled.
8. Set the **Edge** to rising, falling, or either.
9. If you are setting a digital or TSP-Line input, select the input **Line** that generates the trigger.
10. To start the measurements, select the Trigger Mode Indicator at the upper right of the screen and select **Initiate Trigger Model**. The trigger model waits for an input on the selected source.

To set up waveform analog triggers:

1. Set the function to DC current, DC voltage, digitize current, or digitize voltage function.
2. If you selected the DC current or DC voltage function:
   a.   Press the **MENU** key.
   b.   Select **Settings**.
   c.   Select a range (the range cannot be set to Auto).
   d.   Set Auto Zero to **Off**.
3. Press the **MENU** key.
4. In the View menu, select **Graph**.
5. Select the **Trigger** tab.
6. Set the Source Event to **Waveform**.
7. Select the type of waveform:
   ▪ Edge: The trigger event occurs when the signal crosses a certain level.
   ▪ Pulse: The trigger event occurs when a pulse satisfies the amplitude, polarity, and pulse width requirements that you specify.
   ▪ Window: The trigger event occurs when the signal enters or exits a window that is defined by low and high signal levels.
8. Set the length of the **Delay** that occurs before each measurement. 0 sets no delay.
9. Set the **Position**. The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled.
10. Turn **High Frequency Rejection** On or Off as needed.
11. See the following procedures to set the remaining settings for the type of waveform.

NOTE

If you have a fast cyclic signal, the trigger may occur before the instrument can gather sufficient pretrigger data. If this occurs, you will see less pretrigger data than expected. However, the correct amount of posttrigger data will be collected.

Select the options for the Edge waveform:

1. Set the **Level** to the signal level that generates the trigger event.
2. Set the **Slope** to rising or falling. Rising causes a trigger event when the analog signal trends from below the analog signal level to above the level. Falling causes a trigger event when the signal trends from above to below the level.
3. To start the measurements, select the Trigger Mode Indicator at the upper right of the screen and select **Initiate Trigger Model**. The trigger model waits for an edge that meets the criteria.

Select the options for the Pulse waveform:

1. Set the **Level** to the pulse level that generates the trigger event.
2. Set the **Condition** to Greater or Less. This defines if the pulse must be greater than or less than the pulse width before a trigger is generated.
3. Set the **Polarity** to Above Level or Below Level. This determines if the trigger occurs when the pulse is above the defined signal level or below the defined signal level.
4. Set the **Pulse Width**. This sets pulse width that generates a trigger event.
5. To start the measurements, select the Trigger Mode Indicator at the upper right of the screen and select **Initiate Trigger Model**. The trigger model waits for a pulse that meets the criteria.

Select the options for the Window waveform:

1. Set the **Low Boundary** value of the window.

2. Set the **High Boundary** value of the window.

3. Set the Direction. Select **Entering** if the analog trigger occurs when the signal enters the window defined by the boundaries. Select **Leaving** if the analog trigger occurs when the signal leaves the window.

4. To start the measurements, select the Trigger Mode Indicator at the upper right of the screen and select **Initiate Trigger Model**. The trigger model waits for a signal that meets the criteria.

Using the Graph tab

You can touch the plot on the Graph tab to zoom in or out on data, display specific data points, or change which area of the graph you are looking at.

Figure 96: Model DMM7510 Graph tab with a group
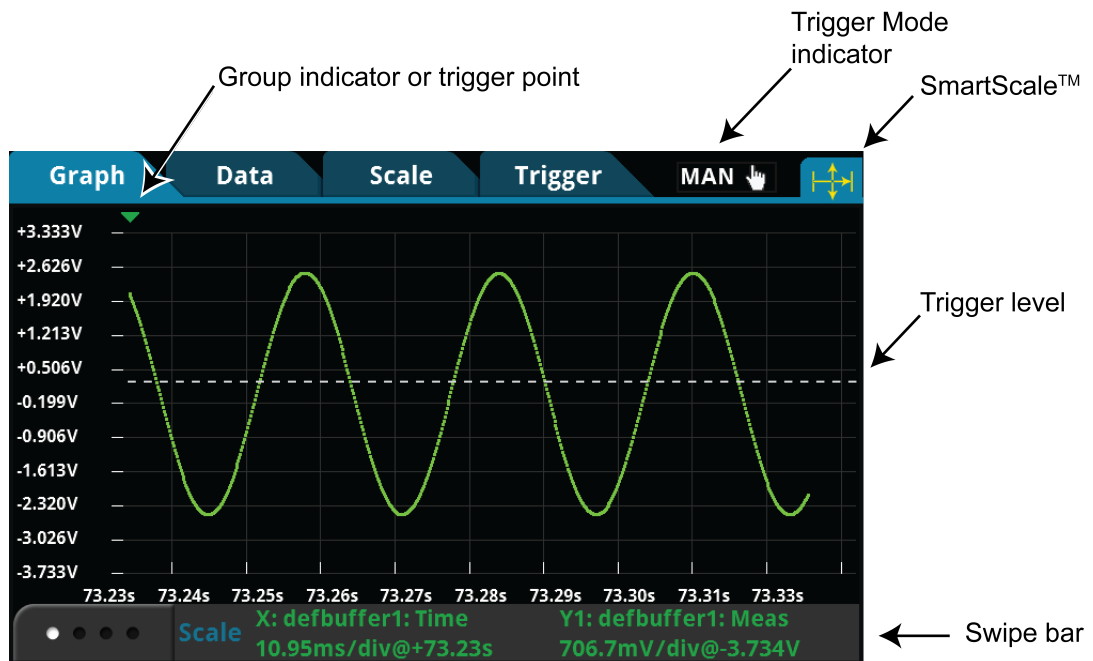
To zoom in on data, use two fingers and flick out. To zoom back out, pinch in. To view earlier data, swipe to the left.

When you manually adjust the scaling of the data, automatic scaling is turned off. To return to automatic scaling, select SmartScale™ in the upper right corner of the Graph tab. When SmartScale is on, the instrument keeps the latest data displayed and determines the best way to scale data based on the data and the instrument configuration (such as the measure count).

To view the values at a specific data point, zoom into a scale where the data point is visible and touch the data point. The Data Point dialog box is displayed with the X and Y values of that point.

If you have the measure count set to more than 1, the graph tracks data as a group. The Group Indicator (shown in the figure above) shows the start of the group.

You can adjust the trigger level by dragging it. The level that is set on the graph is the new level setting. Note that setting the level using remote commands does not affect the graph trigger level.

The timestamp on the X-axis shows the timestamps. As the values of the timestamps become large, the first part of the timestamp is displayed to the left in orange and subsequent digits are displayed on the axis, prefaced by two orange dots (. .).

If the Y-axis displays a ?, there are multiple units in the reading buffer. Clear the buffer to clear the inconsistent units.

You can initiate a trigger model from the Graph screen:

1.   Select the **Trigger Mode** indicator in the indicator bar.
2.   Select **Initiate Trigger Model** from the menu.

You can also press the **TRIGGER** key to initiate a trigger model.

Using the swipe bar

The swipe bar at the bottom of the Graph tab shows which buffers are providing the displayed data, the scale that is used, buffer statistics, and cursors.

The Legend swipe lists each trace and the buffer that is providing the data for that trace.

Figure 97: Graph tab legend
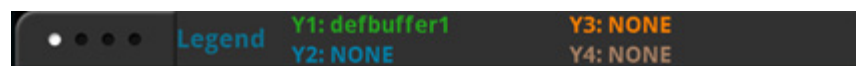
The Scale swipe displays the scale for each trace. Select the trace button at the far left of the swipe to view the scale for other traces.

Figure 98: Graph tab Scale swipe

The Buffer Stats swipe shows the minimum, maximum, average, and standard deviation data for each trace.

Figure 99: Graph tab Buffers Stats swipe
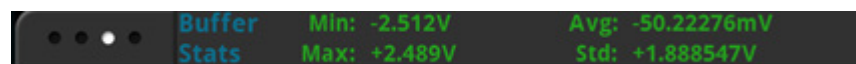
The Cursor swipe allows you to set no, vertical, horizontal, or both cursors. When cursors are displayed, the Cursor swipe displays the values of each cursor and the difference between values between the cursors. Vertical cursors also show values of the data points near the cursor as V1 and V2.

When cursors are displayed, you can drag them to change their positions. You can also move the graph behind the cursors. To move the graph, select a portion of the graph that is not near a cursor and drag. Note that you cannot use the knob to change cursor position.

If you have multiple traces selected, you can select the trace button on the lower left to display data for the cursors as related to the selected trace.

Figure 100: Graph tab Cursor swipe
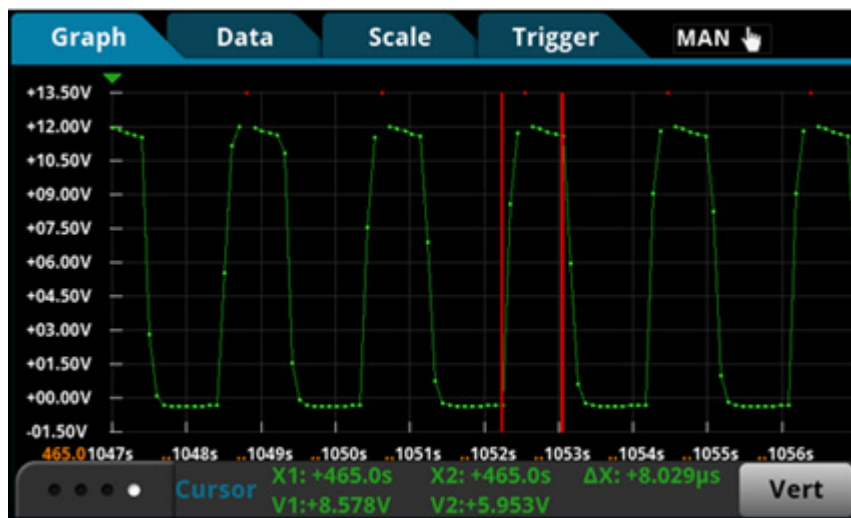
Binning data with the Histogram

The histogram displays data from the active reading buffer in a bar graph with data organized into bins. Data is binned until the statistics are cleared, the buffer is cleared, or a change to the histogram scale settings occurs.

The legend at the bottom of the screen displays the reading buffer, buffer statistics, and the number of bins.

If the top of a bin is red, there is additional data in the bin that is off the screen.

You can change the how the data is displayed:

• Touch the screen with two fingers and pinch or zoom to change the scale of the displayed graph.

• Select a histogram point to display the bin label and count.

• Use the options in the Data and Scale tabs to change how data is displayed.

Setting up the Histogram

To set up the Histogram:

1. Press the **MENU** key.
2. In the View menu, select **Histogram**.
3. Select the **Data** tab.
4. Select the **Bin Buffer**. This reading buffer contains the data that is binned.
5. Select the **Scale** tab.
6. Set the **Minimum Boundary** to the lowest value of data to be binned. Any data below this value is binned in the low outlier bin.
7. Set the **Maximum Boundary** to the highest value of data to be binned. Any data above this value is binned in the high outlier bin.
8. Set the **Number of Bins** to the bins in which to group the data. Two additional outlier bins are added to the number of bins to capture data that is outside the specified boundaries.
9. In most cases, set **Auto Scale** to SmartScale. When SmartScale is selected, the instrument attempts to keep all data in view on the Histogram tab.

NOTE

You can pinch and zoom to change the scale of data. When you adjust the data on the screen, Auto Scale is turned off. You can turn it on in the Scale tab.

10. To clear the Histogram, on the Data tab, select Clear Buffer. This clears the data from the selected reading buffer and the statistics for the buffer, which in turn clears the Histogram.

An example of a histogram tab set for 25 bins is shown in the following figure.

Figure 101: Model DMM7510 Histogram
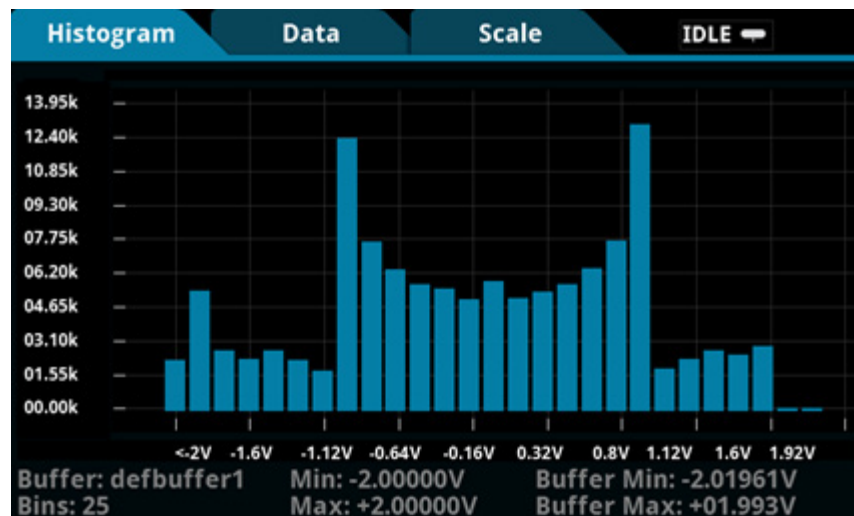
Automatic reference measurements

To ensure the accuracy of readings, the instrument must periodically get new measurements of its internal ground and voltage reference. The time interval between updates to these reference measurements is determined by the integration aperture that is being used for measurements. The Model DMM7510 uses separate reference and zero measurements for each aperture.

By default, the instrument automatically checks the reference measurements whenever a signal measurement is made. If the reference measurements have expired when a signal measurement is made, the instrument automatically makes two more readings, one for the internal ground and one for the voltage reference, before returning the result. This can cause some measurements to take longer than normal.

This additional time can cause problems in test sequences in which measurement timing is critical. To avoid the time that is needed for the reference measurements, you can disable the automatic reference measurements.

When automatic reference measurements are turned off, the instrument may gradually drift out of specification. To prevent inaccurate readings, you can use autozero once to update the autozero information.

Setting autozero

You can enable or disable automatic referencing, or request a one-time refresh of the reference values.

The reference setting is stored with the selected measure function.

To set autozero using the front panel:
1. Press the **FUNCTION** key.
2. Select the measure function.
3. Press the **MENU** key.
4. Under Measure, select **Settings**.
5. For Auto Zero, select **On** or **Off**.
6. If Off is selected, you can select the **Zero Once** option to send a one-time refresh.
7. Select **HOME** to return to the operating display.

To set autozero using SCPI commands:

Refer to the following commands:

* [:SENSe[1]]:<function>:AZERo[:STATe] (on page 6-68)
* [:SENSe[1]]:AZERo:ONCE (on page 6-110)

To set autozero using TSP commands:

Refer to the following commands:

* dmm.measure.autozero.enable (on page 8-129)
* dmm.measure.autozero.once() (on page 8-130)

Saving setups

You can save the present settings and any configuration lists that you have defined for the Model DMM7510 to internal memory or an external USB flash drive.

After the settings are saved, you can recall the settings. You can also set them to be the default settings when the instrument is powered on.

If you are using TSP commands, saved setups are scripts and can be added, modified, and deleted like any other script. See <u>Introduction to TSP operation</u> (on page 7-1) for additional information about working with scripts.

Save a user setup to internal memory

From the front panel:

1. Configure the Model DMM7510 to the settings that you want to save.
2. Press the **MENU** key.
3. Under Scripts, select **Create Setup**. The CREATE SETUP window is displayed.
4. Select **Create**. A keyboard is displayed.
5. Use the keyboard to enter the name of the script.
6. Select the **OK** button on the displayed keyboard. The script is added to internal memory.

Using SCPI commands:

Configure the instrument to the settings that you want to save. To save the setup, send the command:

```
*SAV <n>
```

Where `<n>` is an integer from 0 to 4.

<div style="text-align:center; background:#6b82b4; color:white; font-size:1.4em;">NOTE</div>

In the front-panel script menus, the setups saved with the `*SAV` command have the name `Setup0x`, where *x* is the value you set for `<n>`.

Using TSP commands:

Configure the instrument to the settings that you want to save. To save the setup, send the command:

```
createconfigscript("setupName")
```

Where `setupName` is the name of the setup script that will be created.

Save a user setup to a USB flash drive

From the front panel:
1. Save the user setup to internal memory, as described in <u>Save a user setup to internal memory</u> (on page 2-148).
2. Insert the USB flash drive into the USB port on the front panel.
3. Press the **MENU** key.
4. Under Scripts, select **Manage**. The MANAGE SCRIPTS window is displayed.
5. In the Internal Scripts list, select the script you want to copy to the USB flash drive.
6. Select **>**. The file is transferred to the USB flash drive, and the corresponding filename is displayed in the USB Scripts box.

Using TSP commands:
1. Save the user setup to internal memory, as described in <u>Save a user setup to internal memory</u> (on page 2-148).
2. Insert the USB flash drive into the USB port on the front panel.
3. Send the command:
```
setupName.save("/usb1/USBSetupName")
```
Where *setupName* is the name of the user setup and *USBSetupName* is the name of the file on the USB flash drive. You can use the same name for *setupName* and *USBSetupName*.

Copy a user setup

To copy a user setup from an external USB flash drive to the instrument from the front panel:
1. Insert the USB flash drive into the USB port on the front panel.
2. Press the **MENU** key.
3. Under Scripts, select **Manage**. The MANAGE SCRIPTS window is displayed.
4. In the USB Scripts list, select the script you want to copy from the USB flash drive.
5. Select **<**. The file is transferred to the USB flash drive, and the corresponding filename is displayed in the Internal Scripts box.

Delete a user setup

To remove a user setup from internal memory or the USB flash drive from the front panel:
1.   Press the **MENU** key.
2.   Under Scripts, select **Manage**. The MANAGE SCRIPTS window is displayed.
3.   Under Internal Scripts or USB Scripts, select the name of the script.
4.   Select **Delete**. A confirmation message is displayed.
5.   Select **OK**.

To delete a user setup from internal memory using SCPI commands:

You must overwrite an existing setup with the new setup. See <u>Save a user setup to internal memory</u> (on page 2-148).

To delete a user setup from internal memory using TSP commands:

To delete the setup, send the command:

```
script.delete("setupName")
```

Where `setupName` is the name of the script that will be deleted.

Recall a user setup

You can recall setups from internal nonvolatile memory or a USB flash drive. When you recall a setup, you run a script that restores the instrument to the settings that are saved in that script.

To recall a saved setup from the front panel:
1.   Press the **MENU** key.
2.   Under Scripts, select **Run**.
3.   In the Available Scripts list, select the script you want to recall. USB scripts have the prefix `usb1/`.
4.   Select **Run Selected**.

To recall a user setup from internal memory using SCPI commands:

Send the command:

```
*RCL <n>
```

Where `<n>` is an integer from 0 to 4 that represents the saved script.

To recall a saved setup using TSP commands:

Send the command:

```
setupName()
```

Where `setupName` is the name of the script that contains the setup that was saved with `createconfigscript()`.

Define the setup used when power is turned on

You can select a configuration to be used when power is turned on.

From the front panel:

1. Set the instrument to the settings that you want it to have each time the power is turned on.
2. Press the **MENU** key to open the main menu.
3. Under Scripts, select **Create Setup**. The CREATE SETUP window is displayed.
4. Select **Create**. A keyboard is displayed.
5. Enter the name of the new script, and then select **ENTER** on the keyboard to save it.
6. The instrument saves all present system settings to the script and displays a confirmation message. Click **OK**.
7. Press the **EXIT** key to return to the main menu.
8. Under Scripts, select **Run**. The RUN SCRIPTS window opens.
9. Select the script you just created.
10. Select **Copy to Power Up**.
11. Click **OK** on the confirmation message.

Using a SCPI command:

Send the command:

```
:SYSTem:POSetup <name>
```

Where `<name>` is:

- `RST`: Use the *RST defaults.
- `SAV0`: Use the setup stored at memory location 0
- `SAV1`: Use the setup stored at memory location 1
- `SAV2`: Use the setup stored at memory location 2
- `SAV3`: Use the setup stored at memory location 3
- `SAV4`: Use the setup stored at memory location 4

Using a TSP command:

Save the script that you want to use as the power-on default to be `autoexec`. For example, to save the commands that are presently in the instrument to be the power-on defaults, send the command:

```
createconfigscript("autoexec")
```

Using the event log

The event log records events, which can be errors, warnings, and information reported by the instrument. Through the Event Log menu, you can view these events. You can also specify which events are shown in the event log, which ones are logged, and which ones generate popup messages.

Information provided for each event log entry

Each event log entry includes the following information:

- The date and time when the event occurred in 24-hour time format (MM/DD HH:MM)

- The code number of the event; if you are using a remote interface, you can use this number with the status model to map events to bits in the event registers

- The type of event (separate icons for informational, error, or warning)

- The description of the event

To access an event log listing from the front panel:
1. Press the **MENU** key.
2. Under System, select **Event Log**.
3. Select the **System Events** tab. A list of events is displayed.
4. If the events fill the page, you can scroll down to see additional events.
5. To view additional detail about an event, select the event. A dialog box with additional detail is displayed.

Event log settings

You can set which events you can see in the instrument event log, and which events cause a status message indicator to be displayed on the front panel of the instrument. You can also choose whether or not to log all commands the instrument receives in the event log, which can be useful for troubleshooting problems. You can save the contents of the event log to a USB flash drive or clear the event log.

To access event log settings from the front panel:
1. Press the **MENU** key.
2. Under System, select **Event Log**.
3. Select the **Log Settings** tab. A list of settings is displayed.
4. Make the settings as needed.

The options available on this tab are described in the table below.

Settings tab settings	Description
Show Warning	Turns the display of warnings on or off. If you turn this off, the instrument continues to record warnings and display warning popup messages, but does not display them on the System Events tab.
Show Information	Turns the display of information messages on or off. If you turn this off, the instrument continues to record information messages and display popup messages, but does not display them on the System Events tab.
Log Warning	Turns the logging of warnings on or off. If this is turned off, the instrument does not log or display popups for warning messages.
Log Information	Turns the logging of information messages on or off. If this is turned off, the instrument does not log or display popups for information messages.
Log Command	Turns the logging of commands on or off. When logging is turned on, the instrument records the commands that are sent to the instrument. It records commands sent from any interface (the front panel or a remote interface).
Popups	Turns the display of popups on or off. Options are: • Errors: Turn off the display of error popups. • Errors and Warnings: Turn off the display of error and warning popups. • None: Turn off the display of all popups.
Reset Popups	Restores the popups setting to show errors and warnings.
Save to USB	Saves the event log to a .csv file on the USB flash drive. The filename is eventlog.csv.
Clear Log	Clears all entries from the event log.

Effects of errors on scripts

Most errors will not abort a running script. The only time a script is aborted is when a Lua run-time (event code -286, "TSP runtime error") is detected. Run-time events are caused by actions such as trying to index into a variable that is not a table.

Syntax errors (event code -285, "Program syntax") in a script or command will prevent execution of the script or command.

Resets

There are several types of resets in the Model DMM7510.

In general, the terms "reset," "instrument reset," and "system reset" refer to the reset that is performed when you send the `*RST` or `reset()` command, or when you select **MENU > System > Info/Manage > System Reset** from the front panel. It resets most commands to their default values. Refer to the command descriptions for specifics on which commands are reset by system reset and the default values.

The instrument also responds to other types of resets. These resets include:

- **DMM reset:** This reset is only available if you are using the TSP command set. The `dmm.reset()` function resets any commands that begin with `dmm.` to their default values. Refer to [dmm.reset()](#) (on page 8-195).

- **Password reset:** This resets the instrument password to its default value. You can reset the password by pressing the **MENU** key, selecting **Info/Manage** (under System), and selecting **Password Reset**. When you do this, the password returns to the default setting. Refer to Instrument access (on page 3-1).

- **Digital line reset:** This resets digital I/O line values to their factory defaults if you are using the TSP command set. If you are using SCPI, the lines are reset when the system is reset.

- **LAN reset:** This resets the LAN settings and the instrument password to the system default values. To do this reset, insert a straightened paper clip into hole below LAN RESET on the rear panel. For the location of LAN RESET, refer to Rear panel overview (on page 2-6).

- **Status preset:** This resets all bits in the status model. If you are using the SCPI command set, refer to :STATus:PRESet (on page 6-126). If you are using the TSP command set, refer to status.preset() (on page 8-243).

- **Trigger blender reset:** This reset is only available if you are using the TSP command set. Resets some of the trigger blender settings to their factory defaults. Refer to trigger.blender[N].reset() (on page 8-253).

- **Trigger timer reset:** This reset is only available if you are using the TSP command set. Resets trigger timer settings to their default values. Refer to trigger.timer[N].reset() (on page 8-328).

- **TSP-Link line reset:** This reset is only applicable if you are using TSP-Link. Resets some of the TSP-Link trigger attributes to their defaults. Refer to tsplink.line[N].reset() (on page 8-343).

- **TSP-Net reset:** This reset is only applicable if you are using TSP-NET. Disconnects all TSP-Net sessions. Refer to tspnet.reset() (on page 8-353).

Reset the instrument

You can reset many of the instrument settings to their default values. Default values are listed in the command descriptions.

If you are connected to a TSP-Link system, resetting the instrument resets all TSP-Link enabled instruments on the TSP-Link system.

Using the front panel:
1. Press the **MENU** key.
2. Under System, select **Info/Manage**.
3. Select **System Reset**.
4. The commands are reset and a confirmation message is displayed.

Using SCPI commands:

Send the command:

```
*RST
```

Using TSP commands:

Send the command:

```
reset()
```

NOTE

If the instrument is connected to a TSP-Link system and you are using TSP commands, you can reset only the local instrument by sending `localnode.reset()` instead of `reset()`.

Functions and features

In this section:

Instrument access

You can specify that the control interfaces request access before taking control of the instrument. There are several modes of access.

You can set one of the following levels of access to the instrument:

- **Full**: Allows full access for all users from all interfaces
- **Exclusive**: Allows access by one remote interface at a time with logins required from other interfaces
- **Protected**: Allows access by one remote interface at a time with passwords required on all interfaces
- **Lockout**: Allows access by one interface (including the front panel) at a time with passwords required on all interfaces

NOTE

The front-panel is read-only when you are using a remote interface. You can view information and swipe screens without being prompted to leave remote mode. If you attempt to make a change from the front panel while the instrument is controlled from a remote interface, you will be prompted to enter a password to gain access.

When you set access to full, the instrument accepts commands from any interface with no passwords required. You can change interfaces as needed.

When you set access to exclusive, you must log out of one remote interface and log into another one to change interfaces. To use another interface, log out of the present interface before logging into the new interface. You do not need a password with this access.

Protected access is similar to exclusive access, except that you must enter a password when logging in.

When you set access to locked out, a password is required to change interfaces, including the front-panel interface.

Changing the instrument access mode

To change the access mode from the front panel:
1. Press the **MENU** key.
2. Under System, select **Settings**. The SYSTEM SETTINGS menu opens.
3. Press the button next to Access Mode.
4. Select the level of password access control you want to enable.

Using SCPI commands

Send the command that is appropriate for the level of access you want to enable:

```
SYSTem:ACCess FULL
SYSTem:ACCess EXCLusive
SYSTem:ACCess PROTected
SYSTem:ACCess LOCKout
```

Using TSP commands

Send the command that is appropriate for the level of access you want to enable:

```
localnode.access = localnode.ACCESS_FULL
localnode.access = localnode.ACCESS_EXCLUSIVE
localnode.access = localnode.ACCESS_PROTECTED
localnode.access = localnode.ACCESS_LOCKOUT
```

Changing the password

If the instrument is set to the access mode of Protected or Lockout, you must enter a password to change to a new control interface. You can set the password, as described below.

The default password is `admin`.

To change the password from the front panel:
1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Select the button next to Password. A keypad opens.
4. Enter the new password.
5. Select the **OK** button on the displayed keyboard. A verification screen is displayed.
6. Enter the new password.
7. Select the **OK** button on the displayed keyboard. The password is reset.

> ## NOTE
>
> You can reset the password by pressing the **MENU** key, selecting **Info/Manage** (under System), and selecting **Password Reset**. When you do this, the password returns to the default setting.

To change the password using SCPI commands:

```
:SYSTem:PASSword:NEW "<password>"
```

Where `<password>` is the new password.

To change the password using TSP commands:

```
localnode.password = "password"
```

Where *password* is the new password.

Switching control interfaces

When the access mode is set to anything other than Full, you need to log in to the instrument from the new interface before you can change any settings.

If you are changing to the front panel, when you attempt to make a selection, the Display Lockout - Enter Password keypad is displayed. Enter the password and select the **OK** button on the displayed keyboard.

When you change the remote interface, you must send the following command before sending commands:

```
login password
```

Replace *password* with the instrument password.

Ranges

The measurement range determines the full-scale value of the measurement range for the selected measure function. The range also affects the accuracy of the measurements and the maximum signal that can be measured.

You can allow the Model DMM7510 to choose the range automatically or you can select a specific range.

Auto range selects the best range in which to measure the applied signal. If the measurement reaches 105 percent of the present range, the instrument changes the measurement range to the next higher range. The measurement range is changed when a measurement is made. Auto range is not available for the digitize functions.

When you select a specific range, the instrument remains at the value you selected. This option is intended to eliminate the time that is required by the instrument to automatically search for a range. When selecting a measure range, to ensure the best accuracy and resolution, use the lowest range possible that does not cause an overflow event. Note that when you select a fixed range, overrange conditions can occur.

If you set a specific measure range for a function, auto range is turned off for that function and remains off until you re-enable it.

NOTE
You need to set the measure function before setting the measure range. The range value is stored with the measure function.

Selecting the automatic measurement range

Using the front panel:
1. Press **FUNCTION** and select the function.
2. On the Home screen, select the button next to Range. The Range dialog box is displayed.
3. Select **Auto**. The actual range is displayed to the left of the button.

Using a remote interface:
- SCPI commands: Refer to [:SENSe[1]]:<function>:RANGe:AUTO (on page 6-85).
- TSP commands: Refer to dmm.measure.autorange (on page 8-128).

From the front panel:
1. Press **FUNCTION** and select the measure function.
2. On the Home screen, select the button next to Range in the measurement view area. The Range dialog box is displayed.
3. Select the range. The selected value is displayed.

If the instrument displays an overflow message, select a higher range.

Over a remote interface:
- SCPI commands: Refer to [:SENSe[1]]:<function>:RANGe[:UPPer] (on page 6-86).
- TSP commands: Refer to dmm.measure.range (on page 8-170).

Relative offset

When making measurements, you may want to subtract an offset value from a measurement.

The relative offset feature subtracts a set value or a baseline reading from measurement readings. When you enable relative offset, all measurements are recorded as the difference between the actual measured value and the relative offset value. The formula to calculate the offset value is:

Displayed value = Actual measured value − Relative offset value

When a relative offset value is established for a measure function, the value is the same for all ranges for that measure function. For example, if 4 V is set as the relative offset value on the 100 V range, the relative offset value is also 4 V on the 1 V and 100 mV ranges.

On the front panel, when relative offset is enabled, the REL indicator to the right of the measured value is displayed.

A relative offset value is saved for each function. If you change the measure function, the relative offset value is changed to the setting for that measure function.

The relative offset is applied to the measurement after any math functions but before the limit test functions. For more information on the order in which operations are performed, see Displayed measurements (on page 2-131).

> ## ♡ Quick Tip
>
> You can perform the equivalent of relative offset manually by using the mx+b (on page **3-7**) math function. Set m to 1 and b to the value of the offset.

Establishing a relative offset value

You can use the Model DMM7510 to automatically determine the relative offset, or you can assign a specific relative offset value.

Automatically acquiring a relative offset value

When you automatically acquire a relative offset value, the Model DMM7510:

- Makes a new measurement.
- Stores the measurement as the new relative offset level.

Before acquiring the offset, apply the signal that you want to offset the measurement by.

Using the front panel:

1. Press the **FUNCTION** key and select the measure function.
2. Press the **MENU** key.
3. Select **Calculations**.
4. For Rel, select **Acquire**. The relative offset value is displayed to the right.
5. To enable the relative offset, set Rel to **On**.
6. On the SETTINGS swipe screen, select the box next to **Rel**.

♀ Quick Tip
You can also enable or disable the relative offset feature through the SETTINGS swipe screen Rel option.

Using a remote interface:

*   SCPI commands: Refer to [:SENSe[1]]:<function>:RELative:ACQuire (on page 6-90) and [:SENSe[1]]:<function>:RELative:STATe (on page 6-92).
*   TSP commands: Refer to dmm.measure.rel.acquire() (on page 8-174) and dmm.measure.rel.enable (on page 8-175).

Setting a relative offset value

You can set a specific relative offset value using the remote commands. This option is not available through the front panel.

Over a remote interface:

*   SCPI commands: Refer to [:SENSe[1]]:<function>:RELative (on page 6-88) and [:SENSe[1]]:<function>:RELative:STATe (on page 6-92).
*   TSP commands: Refer to dmm.measure.rel.level (on page 8-176) and dmm.measure.rel.enable (on page 8-175).

Calculations that you can apply to measurements

The Model DMM7510 allows you to apply the following math operations to the measurement:

*   mx+b
*   percent
*   reciprocal (1/X)

Math calculations are applied to the input signal after relative offset and before limit tests. For more detail on the order of operations, see Order of operations (on page 4-16).

Math operations apply to the selected measure function. If you change the measure function, the math operation for that function becomes active.

NOTE
Changing math functions does not clear the reading buffer, which can result in mixed units in the reading buffer. If you are graphing, this can cause ? to be displayed in the Y-axis. Clear the reading buffer to remove the mixed units.

mx+b

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

mx + b = Y

Where:

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

When the mx+b math operation is active, the unit of measure for the front-panel readings is **X** and the MATH indicator is displayed to the right of the measurement. You cannot change this units designator.

Set the relative offset using mX+b

You can use the mX+b function to manually establish a relative offset value. To do this, set the scale factor (m) to 1 and set the offset (b) to the offset value. Each subsequent reading will be the difference between the actual input and the offset value.

Percent

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

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

Where:

Percent = The result

Input = The measurement (if relative offset is being used, this is the relative offset value)

Reference = The user-specified constant

The result of the percent calculation is positive when the input is more than the reference. The result is negative when the input is less than the reference.

When the percent operation is active, the unit of measure for the front-panel readings is **%** and the MATH indicator is displayed to the right of the measurement. You cannot change the unit designator.

Reciprocal (1/X)

You can set math operation to reciprocal to display the reciprocal of a reading.

The reciprocal is 1/X, where X is the reading. If relative offset is on, the 1/X calculation uses the input signal with the relative offset applied.

Example:

Assume the normal displayed reading is 002.5000 Ω. The reciprocal of resistance is conductance. When the reciprocal math function is enabled, the following conductance reading is displayed:

```
0.400000
```

When the reciprocal math operation is active, the unit of measure for the front-panel readings is 1/x and the MATH indicator is displayed to the right of the measurement. You cannot change this units designator.

Setting percent math operations

From the front panel:
1. Press the **FUNCTION** key and select the measure function.
2. Press the **MENU** key.
3. Under Measure, select **Calculations**.
4. Select **Config**.
5. For Math Format, select **Percent**.
6. For Zero Reference, select the percent reference.
7. Select **OK**.
8. Press the **HOME** key to view the measure with the percent math format applied.

Over a remote interface:

- SCPI commands: Refer to :CALCulate[1]:<function>:MATH:FORMat (on page 6-28) and :CALCulate[1]:<function>:MATH:PERCent (on page 6-32).

- TSP commands: Refer to dmm.measure.math.format (on page 8-163) and dmm.measure.math.percent (on page 8-166).

Setting mx+b math operations

From the front panel:
1. Press the **FUNCTION** key and select the measure function.
2. Press the **MENU** key.
3. Under Measure, select **Calculations**.
4. For Math, select **On**.
5. Select **Config**.
6. For Math Format, select **mx+b**.
7. For m(Scalar), set the **m** value.
8. For b(Offset), set the **b** value.
9. Select **OK**.
10. Press the **HOME** key to view the measure with the mx+b math format applied.

Over a remote interface:
*   SCPI commands: Refer to :CALCulate[1]:<function>:MATH:FORMat (on page 6-28), :CALCulate[1]:<function>:MATH:MMFactor (on page 6-30), and :CALCulate[1]:<function>:MATH:MBFactor (on page 6-29).
*   TSP commands: Refer to dmm.measure.math.format (on page 8-163), dmm.measure.math.mxb.mfactor (on page 8-165), and dmm.measure.math.mxb.bfactor (on page 8-164).

Setting reciprocal math operations

From the front panel:
1. Press the **FUNCTION** key and select the measure function.
2. Press the **MENU** key.
3. Under Measure, select **Calculations**.
4. For Math, select **On**.
5. Select **Config**.
6. For Math Format, select **Reciprocal**
7. Select **OK**.
8. Press the **HOME** key to view the measure with the mx+b math format applied.

Over a remote interface:
*   SCPI commands: Refer to :CALCulate[1]:<function>:MATH:FORMat (on page 6-28).
*   TSP commands: Refer to dmm.measure.math.format (on page 8-163).

Switching math on the SETTINGS screen

Once you set the math operations settings for a measure function, you can turn the math function on or off on the SETTINGS swipe screen.

From the front panel:
1. Select **HOME**.
2. Go the SETTINGS swipe screen.
3. Select the button next to **Math** to change the setting.

Filtering measurement data

Filters allow you to produce one averaged sample from a number of measurements. In situations where you have noise levels that fluctuate above and below the measured signal, this can help you produce more accurate measurements.

The Model DMM7510 has two filter options: repeating average and moving average.

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

If you create test algorithms and you are using the averaging filters, make sure the algorithms clear the filter memory stacks at appropriate times to avoid averaging an inappropriate set of measurements.

When the filter is turned on, the relative offset is applied to the filtered reading (moving or repeat). Once the relative offset is applied, the next filtered reading has the relative offset applied before it is reported to the instrument. This means that when you use relative offset, the next reading may not be zero.

For example, if the filter size is set to 10, 10 internal measurements are stored. Once the tenth measurement is made, the display or remote interface updates and returns the average of the 10 readings.

For additional information about the order in which math, filters, offsets, and limits are applied, see Order of operations (on page 4-16).

Repeating average filter

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

Moving average filter

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

Note that when the moving average filter is first selected, the stack is empty. When the first measurement is made, it is copied into all the stack locations to fill the stack. A true average is not produced until the stack is filled with new measurements.

For example, if the filter size is four, the first measurement is copied to all four stack locations. Therefore, (Reading1 + Reading1 + Reading1 + Reading1)/4. The display and remote interface update after first reading. With each additional measurement, the average updates:

(Reading2 + Reading2 + Reading2 + Reading2)/4

(Reading3 + Reading3 + Reading3 + Reading3)/4

(Reading4 + Reading4 + Reading4 + Reading4)/4

Filter window

The filter window sets the window for the averaging filter that is used for measurements for the selected function.

The noise window allows a faster response time to large signal step changes. A reading that outside the plus or minus noise window fills the filter stack immediately.

If the noise does not exceed the selected percentage of range, the reading is based on an average of reading conversions — the normal averaging filter. If the noise does exceed the selected percentage, the reading is a single reading conversion, and new averaging starts from this point.

Setting up the averaging filter

Using the front panel:

1. Press the **MENU** key.
2. Under Measure, select **Calculations**.
3. For Filter, select **On** to enable filtering.
4. Select **Config**.
5. For the Filter Type, select **Moving** or **Repeat**.
6. For the Filter Count, enter the number of measurements to be made for each averaged measurement sample.
7. For the Filter Window, select a value.
8. Select **OK**.
9. Select **HOME** to return to the home screen to view the measurements with the filter applied.

Using SCPI commands:

To set the averaging filters using SCPI commands, refer to the following command descriptions:

[:SENSe[1]]:<function>:AVERage:COUNt (on page 6-63)
[:SENSe[1]]:<function>:AVERage[:STATe] (on page 6-64)
[:SENSe[1]]:<function>:AVERage:TCONtrol (on page 6-65)
[:SENSe[1]]:<function>:AVERage:WINDow (on page 6-67)

Using TSP commands:

To set the averaging filters using TSP commands, refer to the following command descriptions:

dmm.measure.filter.count (on page 8-144)
dmm.measure.filter.enable (on page 8-145)
dmm.measure.filter.type (on page 8-146)
dmm.measure.filter.window (on page 8-148)

Reading buffers

Reading buffers capture measurements, ranges, and instrument status. The Model DMM7510 has two default reading buffers. You can also create user-defined reading buffers.

Reading buffers provide statistics, including average, minimum, maximum, and standard deviation. If you use SCPI commands over the remote interface, peak-to-peak statistics are also available.

You can perform the following operations on reading buffers from the front panel or the remote interface:

- Configure, store, and recall reading buffers. Only one reading buffer is active when you control buffers from the front panel.
- View reading buffer content.
- Choose to store readings in the default reading buffers or the user-defined reading buffers.
- Save reading buffer content to a USB flash drive.
- Set reading buffers to fill once or fill continuously.
- Change the capacity of reading buffers.
- Delete user-defined reading buffers. You cannot delete `defbuffer1` and `defbuffer2`.
- Clear reading buffers.
- Clear the default reading buffers and delete the user-defined reading buffers by turning the instrument off or sending an instrument reset command.

Getting started with buffers

The following sections provide you with information to help you start using reading buffers. The section provides additional information about accessing the reading buffers with remote commands.

Using default buffers

There are two default buffers, `defbuffer1` and `defbuffer2`.

If you do not select a specific buffer, all readings are stored in `defbuffer1`. If you want to store readings in `defbuffer2`, you need to select it. If you want to store readings in a user-defined buffer, you need to create the buffer. The user-defined buffer is automatically set to be the active buffer. New readings are stored in the active buffer.

Effects of reset and power cycle on buffers

The instrument clears the default buffers when a reset command is sent or when the power is turned off and then turned on again.

The instrument deletes all user-defined buffers when a reset command is sent or when the power is turned off and then turned on again.

The active buffer is cleared when the function is changed using the front panel.

Buffer fill status

There are several different ways to view buffer fill status from the front panel.

As shown in the following figure, the Active buffer indicator (on page 2-14) in the annunciator bar displays buffer fill status and the STATISTICS swipe screen (on page 2-20) displays buffer statistics.

Figure 102: STATISTICS swipe screen
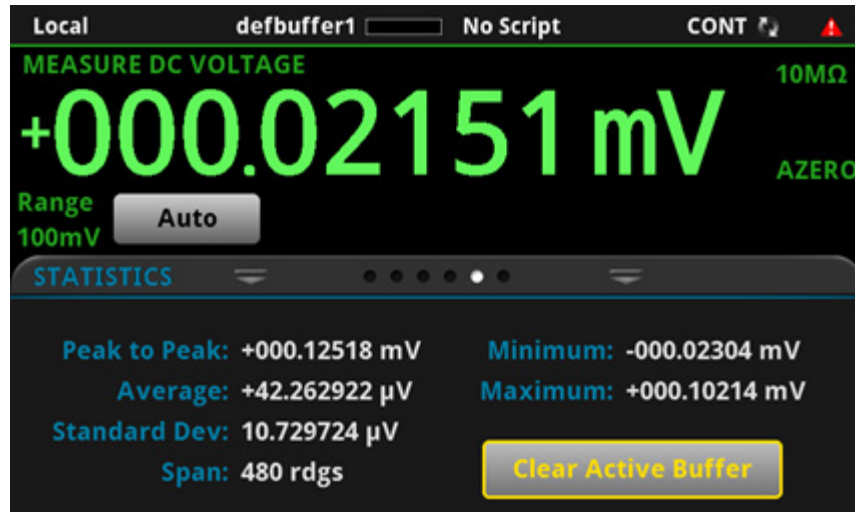
The instrument generates event code 4915, "Attempting to store past capacity of reading buffer," when a buffer that is set to fill once is full.

The MANAGE READING BUFFERS window displays buffer fill status as the Amount Filled, as shown in the following figure.

Figure 103: MANAGE READING BUFFERS
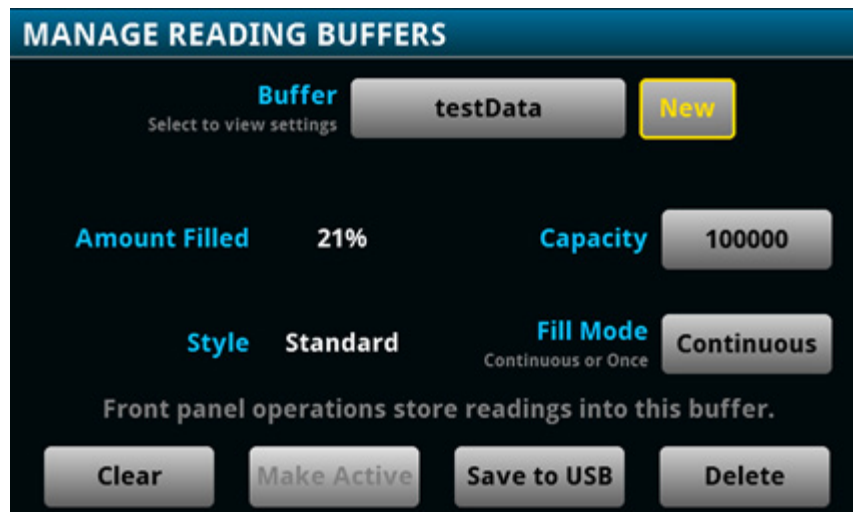
The System Events tab on the System Event Log menu (on page 2-47) displays the following buffer events:

- Event code 4915, "Attempting to store past the capacity of reading buffer," which occurs when a buffer that is set to fill once is full.

- Event code 4916, "The fill status of *bufferVar* is 0% filled."

- Event code 4917, "Reading buffer *bufferVar* is 100% filled."

Timestamps

The measurements in the reading buffers contain timestamps. Readings start at the first entry in the empty reading buffer. Readings are then taken sequentially until the end of the buffer is reached. If the buffer fill mode is continuous, readings wrap to the first entry and fill again. The relative time is taken from the first reading made after a buffer is cleared.

For buffer that fills once, the first entry has a time of 0. For continuous buffers, the lowest timestamp is after the last entry. For example, if you take 150 readings into a buffer with a capacity of 100, the last reading is at entry 50 and the earliest reading is at 151.

Creating buffers

To create a new user-defined reading buffer, you need to provide a name, capacity, and style for the new buffer.

User-defined buffer names must start with an alphabetic character. The names cannot contain any periods or the underscore (_) character. The name can be up to 32 characters long.

There is no fixed limit on the number of user-defined reading buffers you can create. However, you are limited by available memory in the instrument. The overall capacity of all buffers stored in the instrument cannot exceed 11,000,000 readings for standard reading buffers and 27,500,000 for compact reading buffers.

The following topics provide information about using the front panel to create buffers and introduce how to use remote commands to create buffers.

NOTE

The instrument automatically selects the reading buffer you just created as the buffer in which to store front-panel readings.

For additional information about using remote commands for buffer operations, see the following sections of this manual:

- Remote buffer operation (on page 3-27)
- SCPI commands, see TRACe subsystem (on page 6-145)
- TSP commands, see TSP commands (on page 8-7)

Using the front panel to create a user-defined reading buffer:

1. Press the **MENU** key.
2. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.

Figure 104: MANAGE READING BUFFERS window
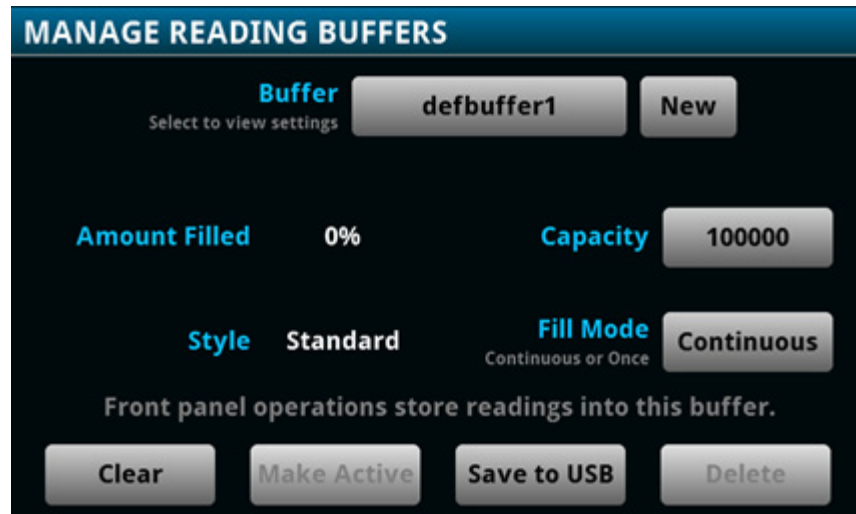
3. Select **New**. A keyboard is displayed.
4. Enter a name for the buffer you are creating, for example, `testData`.
5. Select the **OK** button on the displayed keyboard.
6. The Style dialog box is displayed. You can select:
   - Standard: Store readings with full accuracy with formatting, maximum 11,000,000 readings.
   - Compact: Store readings with reduced accuracy (6.5 digits) with no formatting information, 1 μs accurate timestamp, maximum 27,500,000 readings.
   - Full: Store the same information as standard, plus additional information, such as the ratio component of a DCV ratio measurement

7.   The Initial Capacity window is displayed. Enter the number of readings that the buffer should hold.

Figure 105: Initial Capacity window
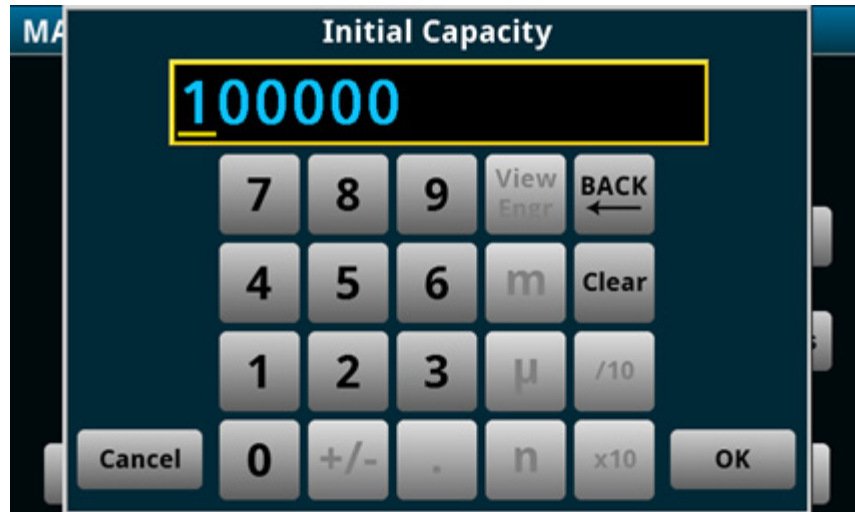
8.   Select **OK**. The MANAGE READING BUFFERS window is displayed, showing the buffer you just created.

Figure 106: MANAGE READING BUFFERS window
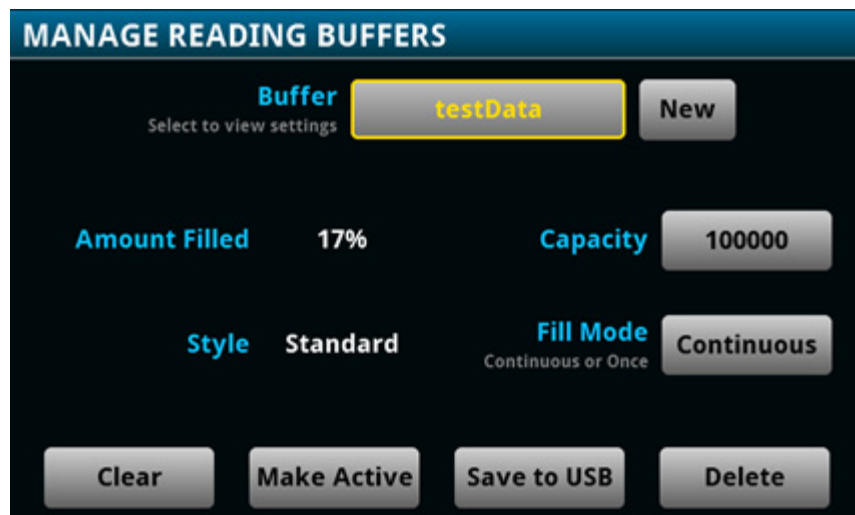
9.   Press the **HOME** key to return to the home screen.

After you create a new reading buffer, the new reading buffer becomes the active buffer.

Using SCPI commands to create a reading buffer:

To create a reading buffer named `testData` with a capacity of 200 readings, send the following command:

```
TRACe:MAKE "testData", 200
```

Using TSP commands to create a reading buffer:

To create a reading buffer named `testData` with a capacity of 200 readings, send the following command:

```
testData = buffer.make(200)
```

Selecting a buffer

The default reading buffer is `defbuffer1`. You can also use a different buffer (`defbuffer2` or a user-defined reading buffer).

When you use remote commands to create buffers, the buffers are available to the system and can be used with any command that takes a buffer parameter. A newly created buffer automatically becomes the active buffer. If the active buffer is deleted, `defbuffer1` becomes the active buffer.

Using the front panel:

1. Press the **MENU** key.
2. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.

Figure 107: MANAGE READING BUFFERS screen
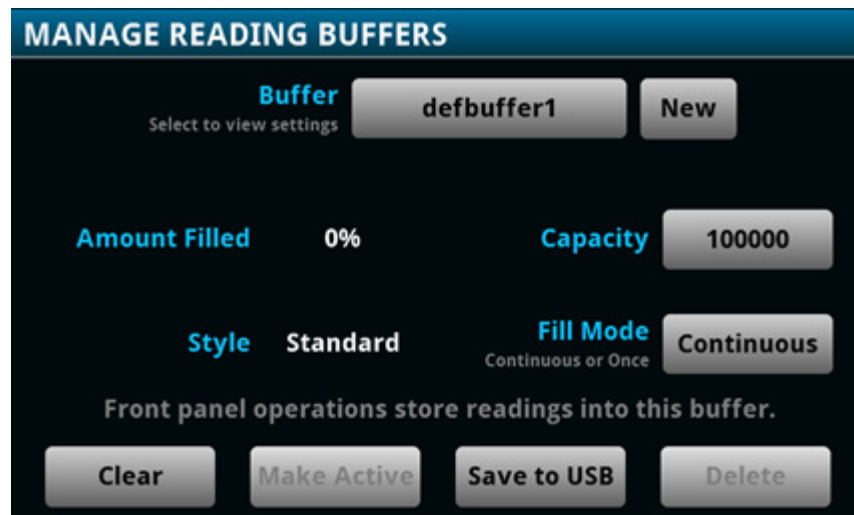
3. Select a reading buffer from the list. For example, select `testData`.

Figure 108: Settings for reading buffer screen
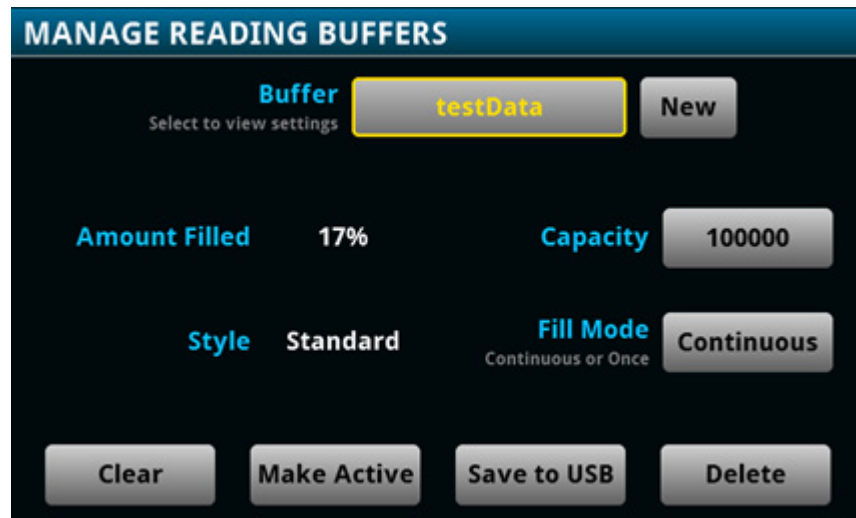
4. Select the **Make Active** button. The "Are you sure" dialog box is displayed.
5. Select **Yes**.

You can also select reading buffers from the active buffer indicator on the Home screen. Refer to Active buffer indicator (on page 2-14) for information about using the indicator to select buffers.

Using SCPI commands to select a reading buffer:

To make a measurement and store the readings in a specific reading buffer, send the command:

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

If you do not specify a buffer name, readings are stored in `defbuffer1`.

An alternative to sending the `:READ? "<bufferName>"` command is to send the command:

```
:TRACe:TRIGger "<bufferName>"
```

The `:TRACe:TRIGger` command stores readings in the specified reading buffer. If no buffer is specified for the parameter, `defbuffer1` is used. To see the readings stored in the buffer after using this command, use the `:FETCh?` command to see the last reading stored in the buffer or the `:TRACe:DATA?` command to see multiple readings from the buffer.

NOTE

When you specify a user-defined reading buffer, you must create the buffer before you can specify it.

To select current as the measurement function, measure current, and return the readings in the `testData` reading buffer, send the following commands:

```
:SENSe:FUNCtion "CURRent"
:READ? "testData"
```

To measure current and store the readings in the `defbuffer2` reading buffer, send the following command:

```
:MEASure:CURRent? "defbuffer2"
```

To measure voltage and store the readings in the `defbuffer2` reading buffer, send the following command:

```
:MEASure:VOLTage? "defbuffer2"
```

To measure current and return the relative time and a reading, send the following command:

```
:MEASure:CURRent? "testData", REL, READ
```

Buffer storage is consistent whenever readings are taken. The parameters, such as `REL`, only affect what is included in the response. If you do not include parameters, the command only returns the reading.

Using TSP commands to select a reading buffer:

To make a measurement and store the readings in a specific reading buffer, use the `dmm.measure.read(`*bufferName*`)` function. If you do not specify a buffer when you use the `dmm.measure.read()` function, readings are stored in `defbuffer1`.

If you select another buffer and you want to store additional readings in the previously selected buffer, you have to select the buffer again or the instrument will use the buffer you just selected.

To measure DC voltage and store the readings in the `voltMeasBuffer` reading buffer, send the commands:

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.read(voltMeasBuffer)
```

To measure voltage, store the readings in `voltMeasBuffer`, and print the last reading in the buffer, send the command:

```
print(dmm.measure.read(voltMeasBuffer))
```

To measure DC current, store the readings in `defbuffer1`, and print the last reading in the buffer, send the commands:

```
dmm.measure.func = dmm.FUNC_DC_CURRENT
print(dmm.measure.read())
```

Using the front panel to store readings in the selected buffer

Before you store readings, make sure the correct reading buffer is selected. See Selecting a buffer (on page 3-17) for more information.

To store a reading from the front panel, make a measurement. The buffer-fill indicators light up to indicate that the buffer is filling. Depending on the size of the buffer, the lit indicator may be difficult to observe. When all four indicators are lit, the buffer is completely filled. All of the indicators will not be lit if the number of readings stored is less than the selected buffer capacity.

To stop storing readings in a buffer when you are making continuous readings, select the trigger mode indicator and select the **Manual Trigger Mode**. You can press and hold the **TRIGGER** key for about 3 seconds to display the trigger mode window.

NOTE

Stored readings are lost when the instrument is turned off or reset. Stored readings are also lost when you resize a reading buffer.

Viewing and saving buffer content

You can view the content of buffers from the front panel.

However, the front panel may not be flexible enough for your particular type of data analysis. For further analysis, save the contents of the reading buffer to a USB flash drive. The stored file can be loaded directly into Microsoft® Excel® or another tool. The file contains all of the information the instrument records about each data point in the reading buffer. When you save the buffer data, you may indicate a starting or ending point to save only a portion of the data. If you do not specify a starting and ending point, the entire buffer data is saved. You may also specify how you want the time saved with the time format parameter.

You can append the contents of a reading buffer to a file that is already on the USB flash drive. When you append data, you can specify the starting and ending point in the buffer to save only a portion of the data and time format as you do when you save the buffer.

All readings are saved in the comma-separated value (.csv) file format. This format stores tabular data (numbers and text) in plain-text form. You can import the .csv file into a spreadsheet.

The Model DMM7510 does not check for existing files when you save. Verify that you are using a unique name to avoid overwriting any existing .csv files on the flash drive.

Using the front panel to view the contents of a reading buffer:
1. Press the **MENU** key.
2. Under **Views**, select **Reading Table**. Data for the active reading buffer is displayed.
3. To display data for a different reading buffer, select the buffer.
4. To view a specific data point, select **Jump To** and enter the number of the data point.
5. Press the **HOME** key to return to the home screen.

Using the front panel to save or append buffer content to files:

1. Insert a USB flash drive into the USB port.
2. Press the **MENU** key.
3. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.
4. Select the reading buffer that you want to save. For example, select `testData`. The Settings for testData menu is displayed.

Figure 109: MANAGE READING BUFFERS window
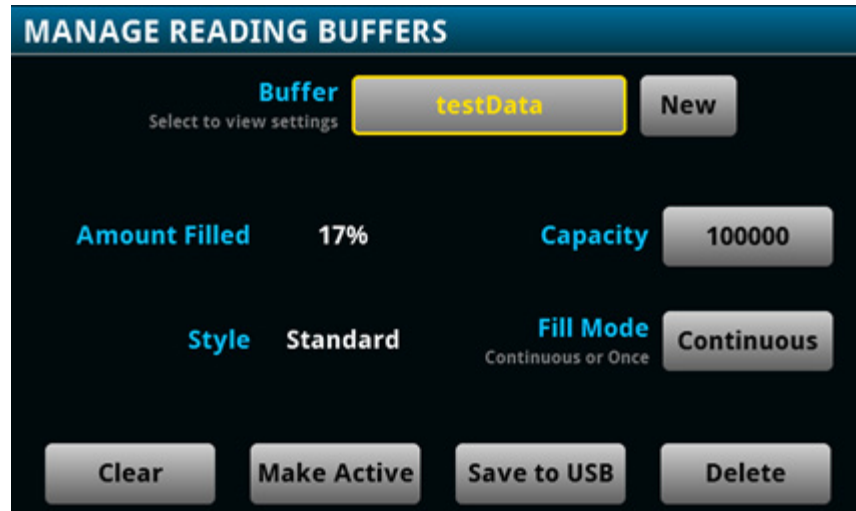
5. Select the **Save To USB** button. A keyboard is displayed.
6. Enter the name of the file in which to save the readings.

NOTE
You only have to enter the name of the file you want to save. It is not necessary to enter the file extension. All files are saved as `.csv` files.

7. Press **OK** on the keyboard.
8. Select **Yes** to confirm saving the file. When the MANAGE READING BUFFERS window is displayed again, the file is saved.
9. Press the **HOME** key to return to the home screen.

Using SCPI commands to save or append buffer content to files:

Before using any of these commands, insert a USB flash drive into the USB port.

To save readings and formatted timestamps from the default buffer to a file named `myData.csv` on a USB flash drive, send the following command:

```
TRACe:SAVE "/usb1/myData.csv", "defbuffer1"
```

To save readings and formatted timestamps from a reading buffer named `testData` to a file named `myData.csv` on a USB flash drive, send the following command:

```
TRACe:SAVE "/usb1/myData.csv", "testData"
```

To append readings and formatted timestamps from a reading buffer named `testData` to a file named `myData.csv` on a USB flash drive, send the following command:

```
TRACe:SAVE:APPend "/usb1/myData.csv", "testData"
```

To append readings and formatted timestamps from a reading buffer named `testData` from point 6 to point 10 in file named `myData.csv` on a USB flash drive, send the following command:

```
TRACe:SAVE:APPend "/usb1/myData.csv", "testData", FORM, 6, 10
```

Using TSP commands to save or append buffer content to files:

Before using any of these commands, insert a USB flash drive into the USB port.

To save readings from the default buffer to a file named `myData.csv` on a USB flash drive, send the following command:

```
buffer.save(defbuffer1, "/usb1/myData.csv")
```

To save readings from a reading buffer named `testData` to a file named `myData.csv` on a USB flash drive, send the following command:

```
buffer.save(testData, "/usb1/myData.csv")
```

To append readings from a reading buffer named *testData* with default time information to a file named `myData.csv` on the USB flash drive, send the following command:

```
buffer.saveappend(testData, "/usb1/myData.csv")
```

Setting reading buffer options

You can adjust the settings for the reading buffers. The settings you can adjust include:

- Buffer capacity: The amount of data the buffer holds
- Fill mode: How the incoming data is managed as the buffer fills

Setting reading buffer capacity

The capacity of a reading buffer determines how many readings the buffer holds. You can change the capacity of reading buffers.

For user-defined buffers, you assign a capacity when you create the reading buffer. For default buffers (`defbuffer1` and `defbuffer2`), the initial buffer size is 100,000 readings.

The following topics describe how to set the reading buffer capacity.

If the reading buffer fill mode is set to fill once, when the buffer reaches capacity, no more readings are made and event code 4915, "Attempting to store past capacity of reading buffer" is displayed.

NOTE

Stored readings and statistics are deleted when you change the capacity of a buffer.

NOTE

When readings are made using a high sample rate and stored into a continuous reading buffer with a capacity of less than 1000 readings, the instrument may not be able to fully process the incoming data before it is overwritten with new data. This can result in gaps in graph traces and the loss of statistics and histogram information. To prevent these problems, increase the buffer capacity or reduce the sample rate.

Using the front panel to set buffer capacity:

NOTE

When you resize a reading buffer, data in the buffer is cleared.

1. Press the **MENU** key.
2. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.

Figure 110: MANAGE READING BUFFERS window
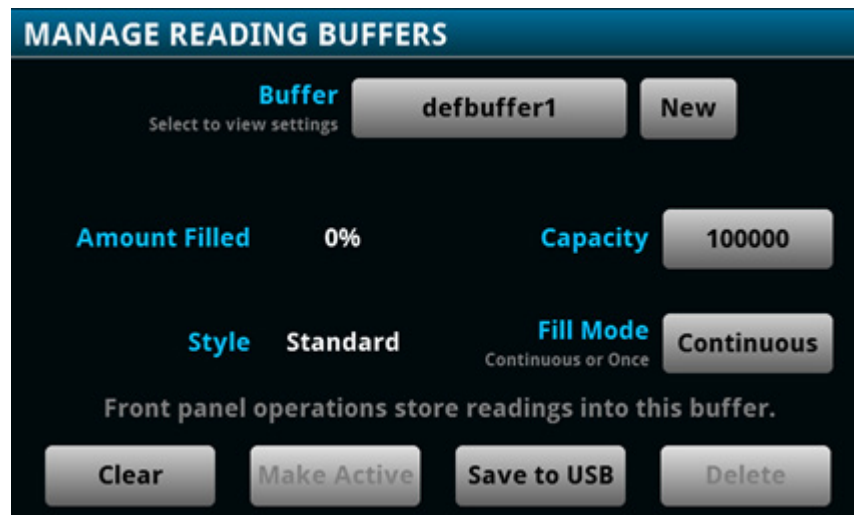
3. Select a reading buffer from the list. For example, select `testData`. The settings for `testData` are displayed.
4. Select the **Capacity** button and enter the new size for the buffer.
5. Select **OK**. The MANAGE READING BUFFERS window is displayed.
6. Press the **HOME** key to return to the Home screen.

Using SCPI commands to set buffer capacity:

To set the `testData` reading buffer to hold 300 readings, send the following command:

```
TRACe:POINts 300, "testData"
```

Using TSP commands to set buffer capacity:

To set the `testData` reading buffer to hold 300 readings, send the following command:

```
testData.capacity = 300
```

Setting the fill mode

The fill mode setting for the reading buffer controls how the incoming data is managed as the buffer fills. You can set the read buffer to:

- **Fill once:** The buffer stops accepting data once it fills to capacity. When the buffer reaches capacity, no more readings are made and event code 4915, "Attempting to store past capacity of reading buffer" is displayed.

- **Fill continuously:** Data fills the buffer normally until the end of the buffer is reached. When the end is reached, the data returns to the beginning of the buffer and overwrites the oldest reading. This is a traditional circular buffer. In this case, the buffer never technically fills.

The following topics describe how to set the reading buffer fill mode.

NOTE

When readings are made using a high sample rate and stored into a continuous reading buffer with a capacity of less than 1000 readings, the instrument may not be able to fully process the incoming data before it is overwritten with new data. This can result in gaps in graph traces and the loss of statistics and histogram information. To prevent these problems, increase the buffer capacity or reduce the sample rate.

Using the front panel to set fill mode:

1. Press the **MENU** key.
2. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.

Figure 111: MANAGE READING BUFFERS window
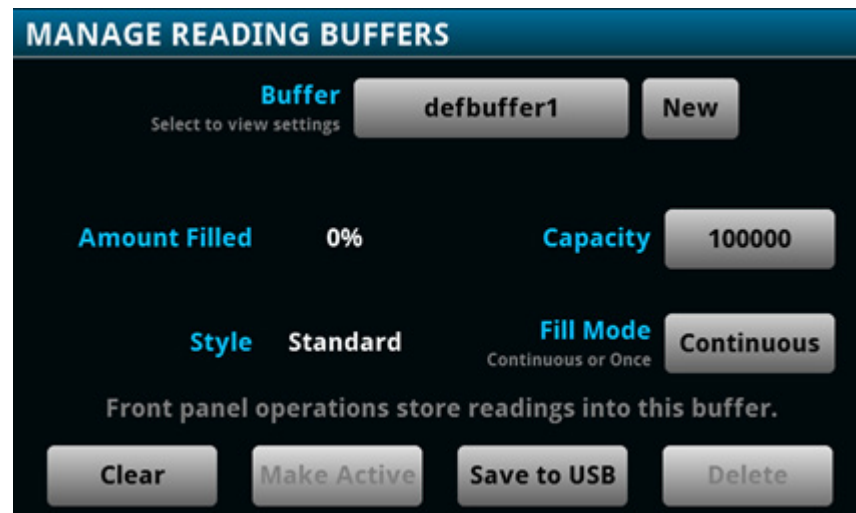
3. Select a reading buffer from the list. For example, select `testData`. The settings for `testData` are displayed.

4. Select the **Fill Mode** option.

5. Press the **HOME** key to return to the home screen.

Using SCPI commands to set the buffer fill mode:

To set the `testData` reading buffer fill mode to continuous, send the following command:

```
TRACe:FILL:MODE CONT, "testData"
```

To set the `defbuffer1` reading buffer fill mode to fill once, send the following command:

```
TRACe:FILL:MODE ONCE, "defbuffer1"
```

Using TSP commands to set a buffer fill mode:

To set the `testData` reading buffer fill mode to continuous, send the following command:

```
testData.fillmode = buffer.FILL_CONTINUOUS
```

To set the `defbuffer1` reading buffer fill mode to fill once, send the following command:

```
defbuffer1.fillmode = buffer.FILL_ONCE
```

To print the `defbuffer1` fill mode setting, send the following command:

```
print(defbuffer1.fillmode)
```

Where a return of `0` indicates the buffer is set to fill once and a return of `1` indicates the buffer is set to fill continuously.

Clearing buffers

You can clear all readings and statistics from buffers.

The following topics provide information about using the front panel to clear buffers and provide an introduction to using remote commands to clear buffers.

Using the front panel to clear a reading buffer:
1. Press the **MENU** key.
2. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.
3. Select a reading buffer from the list. For example, select `testData`.

Figure 112: MANAGE READING BUFFERS window
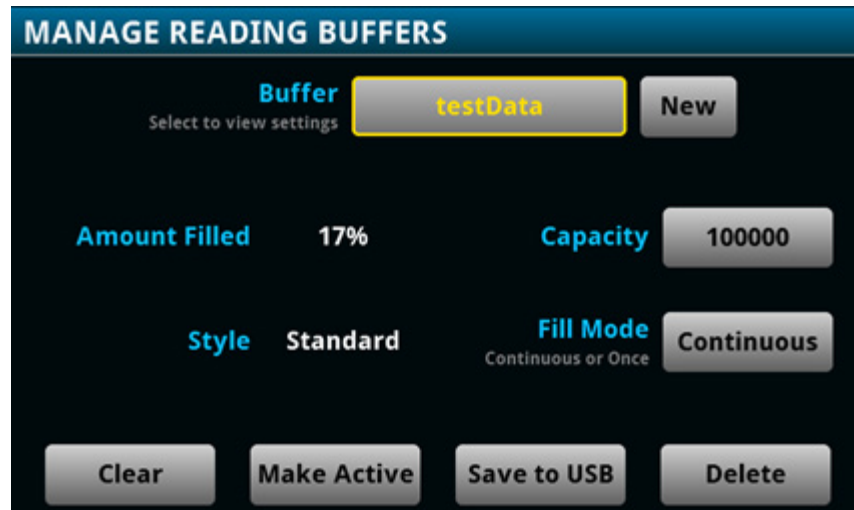
4. Select **Clear** to clear the buffer.
5. A confirmation message is displayed. Select **Yes**.
6. Press the **HOME** key to return to the home screen.

Using SCPI commands to clear a buffer:

To clear a user-defined buffer named `testData`, send the following command:

```
TRACe:CLEar "testData"
```

Using TSP commands to clear a buffer:

To clear a user-defined buffer named `testData`, send the following command:

```
testData.clear()
```

Deleting buffers

If you want to save the readings in a buffer before deleting the buffer, save the buffer to a USB flash drive. See <u>Viewing and saving buffer content</u> (on page 3-20) for details.

You cannot delete the default buffers `defbuffer1` or `defbuffer2`. However, the data in the default buffers is lost when the instrument is reset or the power is turned off.

Using the front panel to delete a reading buffer:
1. Press the **MENU** key.
2. Under Measure, select **Reading Buffers**. The MANAGE READING BUFFERS window is displayed.
3. Select **Buffer**.
4. Select the buffer to be deleted.
5. Select **Delete** to delete the buffer.
6. When the "Are you sure you want to delete testData" prompt is displayed, select **Yes**.

Using SCPI commands:

To delete a user-defined buffer named `testData`, send the following command:

```
:TRACe:DELete "testData"
```

Using TSP commands:

To delete a user-defined buffer named `testData`, send the following command:

```
buffer.delete(testData)
```

<div style="background:#6677aa;color:white;text-align:center;">NOTE</div>

Do not set the buffer name to `nil` to delete it. To cleanly delete the buffer from the instrument, use the `buffer.delete()` command.

Remote buffer operation

You can control the Model DMM7510 buffers through a remote interface using SCPI or TSP remote commands.

This section provides a summary of some of the remote commands available to control and access data stored in buffers; however, this section does not describe all of the available commands. See the following sections for command descriptions:

- For information about SCPI commands, see the SCPI command reference (on page 6-1)
- For information about TSP commands, see the TSP command reference (on page 8-1)

Storing data in buffers

Using SCPI commands:

The table below lists the SCPI commands that you use for data storage.

Command	Description
`:TRACe:MAKE`	This command creates a user-defined reading buffer. You cannot use this command on the default buffers. See Creating buffers (on page 3-14). Also see :TRACe:MAKE (on page 6-152).
`:TRACe:SAVE`	This command saves data from the specified reading buffer to a USB flash drive. See :TRACe:SAVE (on page 6-156).
`:TRACE:SAVE:APPend`	This command appends data from the reading buffer to a file on the USB flash drive. See :TRACE:SAVE:APPend (on page 6-158).
`:TRACe:POINts`	This command reads the number of readings a buffer can store. This allows you to change the number of readings the buffer can store. See :TRACe:POINts (on page 6-155).
`:TRACe:CLEar`	This command clears all readings and statistics from the specified buffer. See Clearing buffers (on page 3-25). Also see :TRACe:CLEar (on page 6-146).
`:TRACe:STATistics:CLEar`	This command clears the statistical information associated with the specified buffer. This command does not clear the readings.
`:TRACe:FILL:MODE`	This command determines if a reading buffer is filled continuously or is filled once and stops. See :TRACe:FILL:MODE (on page 6-150).
`:TRACe:LOG:STATe`	This command indicates whether the reading buffer should log informational events. See :TRACe:LOG:STATe (on page 6-151).
`:TRACe:ACTual?`	This command contains the number of readings in the specified buffer. See :TRACe:ACTual? (on page 6-145)
`:TRACe:DELete`	This command deletes a buffer. See :TRACe:DELete (on page 6-149).

Using TSP commands:

⚠ CAUTION
Once you create a reading buffer using TSP commands, if you use that buffer name for another buffer or variable, you can no longer access the original buffer.

The table below lists the TSP commands that you use for data storage.

Command	Description
`buffer.clearstats()`	This function clears all statistics from the specified buffer. This function does not clear the readings. See buffer.clearstats() (on page 8-15).
`buffer.delete()`	This function deletes a user-defined reading buffer. See buffer.delete() (on page 8-16).
`buffer.make()`	This function creates a user-defined reading buffer. You cannot use this command on the default buffers. See Creating buffers (on page 3-14). Also see buffer.make() (on page 8-18).
`buffer.save()`	This function saves data from the specified reading buffer to a USB flash drive. See buffer.save() (on page 8-20).
`buffer.saveappend()`	This function appends data from the reading buffer to a file on the USB flash drive. See buffer.saveappend() (on page 8-21).
`bufferVar``.capacity`	This attribute reads the number of readings a buffer can store. This allows you to change the number of readings the buffer can store. See bufferVar.capacity (on page 8-25).
`bufferVar``.clear()`	This function clears all readings and statistics from the specified buffer. See Clearing buffers (on page 3-25) and bufferVar.clear() (on page 8-26).
`bufferVar``.fillmode`	This attribute determines if a reading buffer is filled continuously or is filled once and stops. See bufferVar.fillmode (on page 8-29).
`bufferVar``.logstate`	This attribute indicates whether the reading buffer should log informational events. See bufferVar.logstate (on page 8-32).
`bufferVar``.n`	This attribute contains the number of readings in the specified reading buffer. See bufferVar.n (on page 8-33).

Accessing the data in buffers

Using SCPI commands:

To access a buffer, include the buffer name in the respective command. For example, the following commands:

- Create a buffer named `testData` to store 100 readings

- Set the instrument to make 5 readings for all measurement requests

- Make the readings and store them in the buffer

- Return five readings (including the measurement and relative time) from the user-defined buffer named `testData`

```
TRAC:MAKE "testData", 100
SENS:COUN 5
TRAC:TRIG "testData"
TRAC:DATA? 1, 5, "testData", READ, REL
```

Using TSP commands:

A reading buffer is based on a Lua table. When you use TSP commands, the measurements themselves are accessed by ordinary array notation. If `rb` is a reading buffer, the first measurement is accessed as `rb[1]`, the ninth measurement as `rb[9]`, and so on. The additional information in the table is accessed as additional members of the table.

To access a buffer, include the buffer name in the respective command. For example, the following commands:

- Create a buffer named `testData` to store 100 readings
- Set the instrument to make 5 readings for all measurement requests
- Make the readings and store them in the buffer
- Return five readings (including the measurement and relative time) from the user-defined buffer named `testData`

```
-- Create a buffer named testData to store 100 readings.
testData = buffer.make(100)
-- Set the instrument to make 5 readings and store them in the buffer.
trigger.model.load("SimpleLoop", 5, 0, testData)
-- Make the readings
trigger.model.initiate()
waitcomplete()
-- Read the 5 readings and print them including the measurement
-- and relative time for each reading.
printbuffer(1, 5, testData.readings, testData.relativetimestamps)
```

Buffer read-only attributes

Use buffer read-only attributes to access the information contained in an existing buffer.

Using SCPI commands:

The following commands are available for each reading buffer.

Attribute	Description
`:TRACe:ACTual?`	This command contains the number of readings in the specified buffer. See :TRACe:ACTual? (on page 6-145)
`:TRACe:DATA?`	This command contains the readings stored in a specified reading buffer. See :TRACe:DATA? (on page 6-147).
`:TRACe:STATistics:AVERage?`	This command returns average of all readings added to the buffer. See :TRACe:STATistics:AVERage? (on page 6-159)
`:TRACe:STATistics:MAXimum?`	This command returns the maximum reading value added to the buffer. See :TRACe:STATistics:MAXimum? (on page 6-161)
`:TRACe:STATistics:MINimum?`	This command returns the minimum reading value added to the buffer. See :TRACe:STATistics:MINimum? (on page 6-162)
`:TRACe:STATistics:PK2Pk?`	This command returns the peak-to-peak value of all readings added to the buffer. See :TRACe:STATistics:PK2Pk? (on page 6-163).
`:TRACe:STATistics:STDDev?`	This command returns the standard deviation of all readings added to the buffer. See :TRACe:STATistics:STDDev? (on page 6-163).

Using TSP commands:

See [printbuffer()](#) (on page 8-229) for a list of available attributes.

Reading buffer time and date values

Time and date values are represented as a number of UTC seconds since 12:00 a.m. Jan. 1, 1970. Use the following TSP commands to return values in the following formats:

- Hours and minutes: [bufferVar.times](#) (on page 8-39)
- UTC seconds: [bufferVar.seconds](#) (on page 8-36)
- Month, day, year, format, or to access the timestamp table: [bufferVar.dates](#) (on page 8-27)

Reading buffer for . . . do loops

The following TSP examples illustrate the use of for . . . do loops when recalling data from a reading buffer called `mybuffer`. The following code may be sent as one command line or as part of a script. Example outputs follow the line of code. Also see the [printbuffer()](#) (on page 8-229) command.

This example loop uses the `printbuffer()` command to show the reading, units, and relative timestamps for all readings stored in the reading buffer. The information for each reading (reading, units, and relative timestamps) is shown on a single line with the elements comma-delimited.

```
for x = 1, mybuffer.n do
   printbuffer(x,x,mybuffer, mybuffer.units, mybuffer.relativetimestamps)
end
```

Example comma-delimited output of above code:

```
-1.5794739960384e-09, Amp DC, 0
-1.5190692453926e-11, Amp DC, 0.411046134
-2.9570144943758e-11, Amp DC, 0.819675745
-2.9361919146043e-11, Amp DC, 1.228263492
-3.0666566508408e-11, Amp DC, 1.636753752
-4.0868204653766e-11, Amp DC, 2.034403917
```

The following loop uses the `print` command instead of the `printbuffer` command. This loop shows the same information described in the previous example (reading, units, and relative timestamps for all readings stored in the buffer). However, because the `print()` command is used instead of `printbuffer()`, each line is tab-delimited (rather than comma-delimited) to produce a columnar output, as shown below:

```
for x = 1, mybuffer.n do
   print(mybuffer.readings[x], mybuffer.units[x], mybuffer.relativetimestamps[x])
end
```

Example columnar-delimited output of above code:

```
-1.5794739960384e-09 Amp DC        0
-1.5190692453926e-11 Amp DC        0.411046134
-2.9570144943758e-11 Amp DC        0.819675745
-2.9361919146043e-11 Amp DC        1.228263492
-3.0666566508408e-11 Amp DC        1.636753752
-4.0868204653766e-11 Amp DC        2.034403917
```

Saving front-panel settings into a macro script

You can save some settings made through the front panel into a macro script that you can save and run later.

The settings that are saved include any settings made through:

- Measure menu QuickSet, Measure, Calculations, Config Lists, and Reading Buffers
- Trigger menu options Templates and Configure
- The Graph Trigger tab
- System Communications
- Time and date
- Auto calibration schedule options

It also saves the reading format, access mode, and system reset settings.

Macro scripts are limited to 10 kB.

Recording a macro script

To record a macro script:
1. Press the **MENU** key.
2. Under Scripts, select **Record**.
3. Select the **Start Macro** button.
4. Make the settings that you want to record.
5. Press the **MENU** key.
6. Under Scripts, select **Record**.
7. Select the **Stop Macro** button. The Macro Script Name dialog box is displayed.
8. Enter a name for the script.
9. Select the **OK** button.

♀ Quick Tip
You can also stop or cancel recording from the home screen. Select the **Recording** indicator in the indicator bar.

After you create a macro script, you can use the other Scripts menu options to run and manage scripts. Refer to for information on the options.

Configuration lists

Instrument configuration

An instrument configuration is a collection of settings that can be applied to the instrument.

Active setting

At any given time, the instrument is operating using its active settings. For example, if you set the measure NPLC to 1.0, the active NPLC setting is 1.0.

Active state

At any given time, the complete set of active settings of the instrument is the active state. These active settings can be subdivided into the following groups:

- Measure and digitize settings
- General settings

The active state of the instrument changes when:

- You use the front panel of the instrument to change settings.
- You send commands to the instrument that change settings.
- You use a configuration list to recall measure settings.
- You run a configuration script (TSP or front panel) or use `*RCL` (SCPI) to recall all instrument settings.

When you create a new configuration list, it is important to remember that all instrument settings are included in its active state, not just the specific settings that changed immediately before setting up the configuration list.

What is a configuration list?

A configuration list is a list of stored settings for the measure or digitize function. You can restore these settings to change the active state of the instrument.

Configuration lists allow you to record the function settings of the instrument, store them, and then return the instrument to those settings as needed.

You can recall configuration lists from the front panel, using remote commands, or as part of a trigger model.

If you want to use the same configuration list on multiple Model DMM7510 instruments, you have to recreate it on each instrument. You can do this using one of the following methods:

- Define the commands to create the configuration list, then send the commands to multiple instruments.
- Create a user-defined saved setup and run it on the other instruments.

Configuration lists and the trigger model

If you think of the trigger model as the execution engine that makes the instrument do things, configuration lists provide a database of stored settings that the trigger model can recall to change the settings of the instrument at any time during trigger model execution. Refer to Trigger model (on page 3-72) for more information.

What is a configuration point?

A configuration point contains a copy of all instrument measure active settings at a specific point in time. You store configuration points in a specific configuration list. Only the amount of available memory limits the number of configuration points that you can store in a configuration list. Lists may exceed 100,000 points.

Each configuration point is identified by an index. To overwrite an existing point, you can provide the index when you store the configuration point. Otherwise, the instrument appends the configuration point to the end of the list. Configuration points are chronologically numbered with the index starting from 1.

The first time you create a configuration point and store into it, the instrument stores the active settings to configuration point 1. Each time you store another set of active settings to the same list, the instrument creates a new configuration point and appends it to the list using the next chronological index.

You can use the index to identify a specific configuration point and perform operations on it when necessary.

Although you can specify a specific configuration point index when you store active settings to a configuration list, this is only necessary if you wish to overwrite an existing point. Normally, you can build up the configuration points in a configuration list by appending (no index specified) subsequent configuration points to the list.

If you only store one configuration point to a list, the list consists of configuration point 1.

What settings are stored in a configuration list?

Specific instrument settings that affect measurement are stored in a configuration list. The same settings are recalled to overwrite the active state when you recall a configuration list.

The first time you store a configuration list, the instrument stores the active settings to configuration point 1. Each time you append a configuration point to the configuration list, the instrument saves the active settings to a configuration point.

When you recall a configuration point on the list, the instrument restores the settings to the values that were stored. The recall operation overwrites the active settings with the stored settings.

You can only recall the settings from one configuration point at a time.

To see the actual values for the settings that are saved to a configuration point, refer to Viewing configuration list contents (on page 3-38).

Creating, storing, and performing operations on configuration lists and points

To create a configuration point, you need to:

*   Create a new configuration list and give it a name or use a specific configuration list that already exists on the instrument
*   Configure the instrument with the settings that you want to store in a configuration point
*   Store the active settings into a configuration point on the specified configuration list

After you store configuration points to a configuration list, you can do the following operations on a specific configuration point:

*   Recall a configuration point and restore the stored settings to the active state
*   View the contents of a configuration point
*   Delete a configuration point or delete the entire configuration list

You can work with configuration lists from the front panel (see <u>Using the front panel for configuration list operations</u> (on page 3-34)) or by using remote commands (see <u>Using remote commands for configuration list operations</u> (on page 3-39)).

Using the front panel for configuration list operations

This section describes how to store the active settings to a specific point on a configuration list.

See <u>What is a configuration point?</u> (on page 3-33) for information about how the instrument adds configuration points to configuration lists before reading this section.

The following topics provide information to:

*   Create an example measure configuration list named `MyMeasList`.
*   Store two example configuration points on `MyMeasList`. The following table provides information about specific settings for each configuration point.
*   View the contents of a specific configuration point.
*   Recall a configuration point from `MyMeasList`.

Front panel configuration list menu overview

To display the configuration list menus from the main menu:
1. Press the **MENU** key.
2. Under Measure, select **Config Lists**. The MEASURE CONFIGURATION LISTS screen is displayed. Use this menu for operations on measure configuration lists.

Configuration list menu selections:

The following figure shows an example MEASURE CONFIGURATION LISTS menu with four configuration indexes.
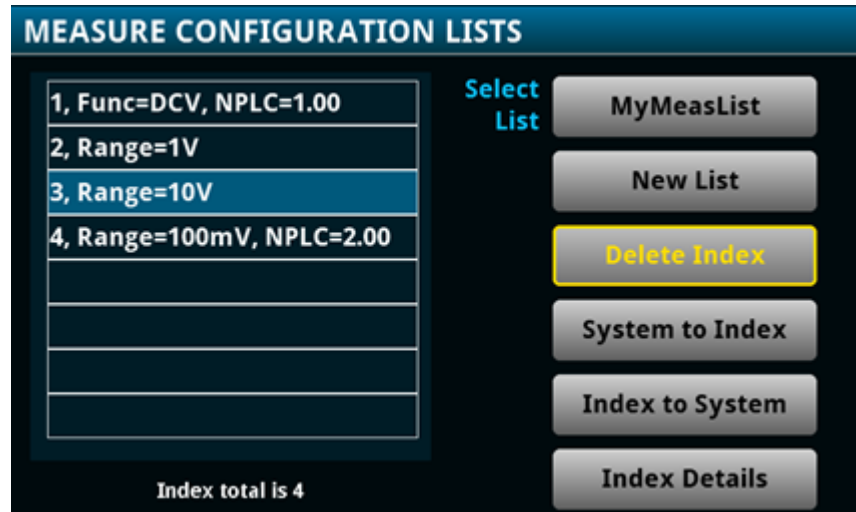
The CONFIGURATION LIST menu contains:

- A scrollable list of configuration points that are stored in the selected configuration list
- A message bar indicating the index associated with the selected configuration point
- Buttons for operations. For descriptions, refer to Measure Config Lists menu (on page 2-35).

Duplicate configuration indexes:

If you store a second configuration index that has the same settings as an index that is already on the configuration list, "No change" is displayed for that index.

Creating a configuration list and giving it a name

This example creates a configuration list named MyMeasList.

Using the front panel to create the configuration list:
1. Press the **MENU** key.
2. Under Measure, select **Config Lists**. The MEASURE CONFIGURATION LISTS screen is displayed.
3. Select **New List**. The keypad is displayed.
4. Enter a name for the configuration list you are creating. For example, MyMeasList.
5. Select the **OK** button on the displayed keyboard. The MEASURE CONFIGURATION LISTS screen is displayed.
6. Select **HOME** to return to the home screen.

Storing configuration point 1

For example, use the following information to store configuration point 1 to `MyMeasList`.

NOTE
You cannot mix digitize and measure functions in a configuration list that is used in a trigger model.

Using the front panel to configure the instrument:

Configure the instrument with the settings you want to store in the configuration point.

1. Press the **HOME** key.
2. Select the **ACV** function.
3. Select **10 V** range.
4. Press the **MENU** key.
5. Under Measure, select **Settings**.
6. Set Detector Bandwidth to **3 Hz**.

Using the front panel to store active settings to configuration point 1:

Store all active measure settings to `MyMeasList` as configuration point 1.

1. Press the **MENU** key.
2. Under Measure, select **Config Lists**. The MEASURE CONFIGURATION LISTS screen is displayed.
3. Choose **Select List**. A menu of available configuration lists is displayed.
4. Select **MyMeasList**.
5. Select **System to Index**. This saves the active system settings to the configuration point. The configuration point is displayed on the list.
6. Select **Index Details** to see some of the settings stored in configuration point 1.
7. Select **HOME** to return to the Home screen.

Figure 113: Configuration List Details example
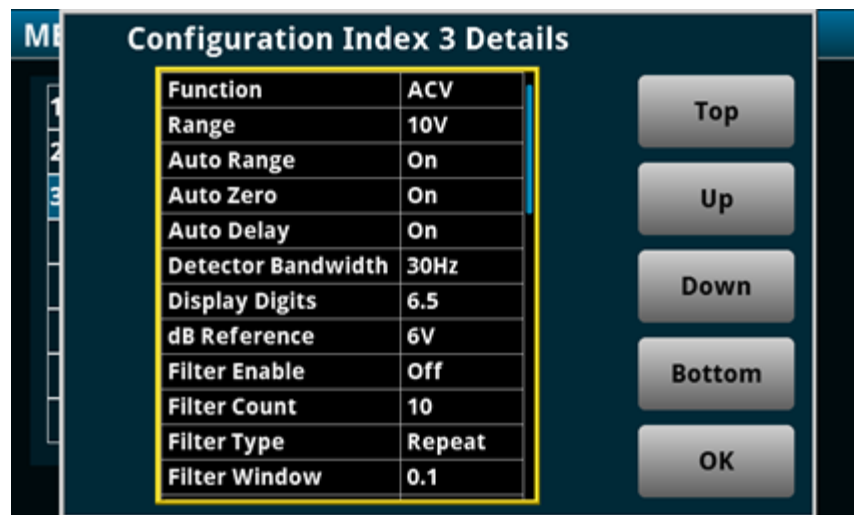
Storing configuration point 2

Refer to the instructions in , and continue as follows to configure the instrument for configuration point 2.

Using the front panel to configure the instrument:

Change the following instrument settings to configure the instrument with the settings you want to save for configuration point 2:

- Set the function to DC voltage
- Set the range to 100 mV
- Set the integration rate to 2 PLC

Using the front panel to store the active settings to configuration point 2:
1. Return to the configuration list menu.
2. Select **MyMeasList**.
3. Select **System to Index**. This saves the active system settings to the configuration point.
4. Select **HOME** to return to the home screen.

Recalling a configuration point

You can recall the settings stored in a specific configuration point in a configuration list.

For example, use the following procedure to recall configuration point 2 from MyMeasList.

Using the front panel to recall a configuration point:
1. Press the **MENU** key.
2. Under Measure, select **Config Lists**. The MEASURE CONFIGURATION LISTS screen is displayed.
3. Choose **Select List**. A menu of available configuration lists is displayed.
4. Select **MyMeasList**. The configuration points in the list display.
5. Select the second configuration point.
6. Select **Index to System**.

Viewing configuration list contents

Use the following procedure to view configuration point 2 from `MyMeasList`.

Using the front panel:

1. Press the **MENU** key.
2. Under **Measure**, select **Config Lists**. The MEASURE CONFIGURATION LISTS screen is displayed.
3. Choose **Select List**. A menu of available configuration lists is displayed.
4. Select **MyMeasList**. The configuration points are displayed.
5. Select the second configuration point.
6. Select **View Details**.

Figure 114: Configuration List Details example
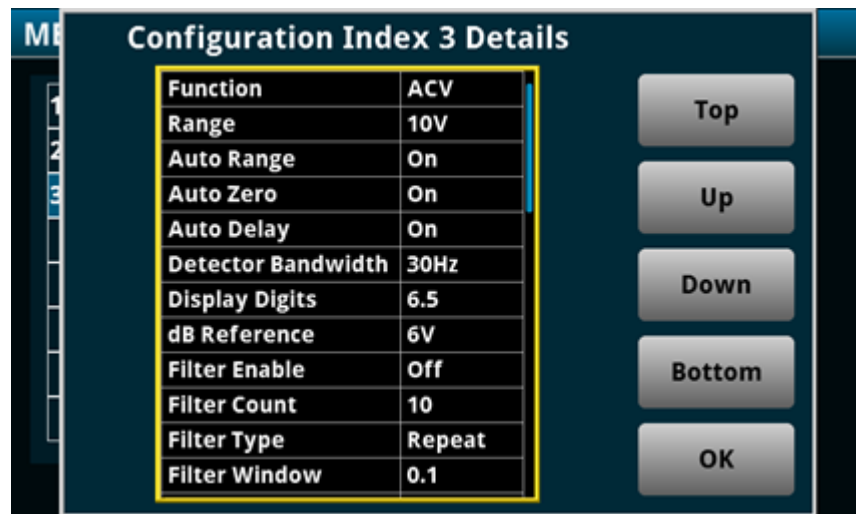
7. When you are finished, select **OK**.

Deleting a configuration point

Use the following procedure to delete a configuration point from `MyMeasList`.

Using the front panel to delete a configuration point:

1. Press the **MENU** key.
   Under **Measure**, select **Config Lists**. The MEASURE CONFIGURATION LISTS screen is displayed.
2. Choose **Select List**. A menu of available configuration lists is displayed.
3. Select **MyMeasList**. The configuration points are displayed.
4. Select the point to be deleted.
5. Select **Delete Index**.

Using remote commands for configuration list operations

You can use the following remote commands to create and maintain configuration lists.

Action	SCPI command TSP command
Create a configuration list	[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-112) dmm.measure.configlist.create() (on page 8-134)
Restore the settings in a configuration list to the instrument	[:SENSe[1]]:CONFiguration:LIST:RECall (on page 6-114) dmm.measure.configlist.recall() (on page 8-137)
View the contents of a configuration list point as TSP commands	[:SENSe[1]]:CONFiguration:LIST:QUERy? (on page 6-113) dmm.measure.configlist.query() (on page 8-135)
Delete a configuration list or an index point in a configuration list	[:SENSe[1]]:CONFiguration:LIST:DELete (on page 6-112) dmm.measure.configlist.delete() (on page 8-135)
View available configuration lists	[:SENSe[1]]:CONFiguration:LIST:CATalog? (on page 6-111) dmm.measure.configlist.catalog() (on page 8-133)
Determine the number of configuration index points in a configuration list	[:SENSe[1]]:CONFiguration:LIST:SIZE? (on page 6-114) dmm.measure.configlist.size() (on page 8-138)
Save a configuration list	[:SENSe[1]]:CONFiguration:LIST:STORe (on page 6-115) dmm.measure.configlist.store() (on page 8-139)

Saving a configuration list

Configuration lists are lost when you turn the instrument off and turn it on again. Save a configuration list by creating a configuration script (TSP or front panel) or using the *SAV and *RCL commands (SCPI). A configuration script saves the settings of the instrument, including all configuration lists. See Saving setups (on page 2-148) for additional information.

Auto calibration

Automatic calibration removes measurement errors that are caused by the performance drift on the components used in the DMM as a result of temperature and time. Auto calibration improves short-term accuracy of the Model DMM7510. However, you must still perform regular full calibration with metrology equipment to maintain overall accuracy.

To maintain accuracy, run auto calibration when the instrument temperature changes by more than ±5 °C since the last auto calibration. To check the temperature difference, you can view the temperature change on the Calibration menu. You can also use remote commands to retrieve the temperature difference.

The instrument regularly monitors the internal temperature for the voltage, current, 2-wire resistance, 4-wire resistance, diode, temperature, and DC voltage ratio function when autozero is enabled. Temperature checking begins after the warm-up time completes. If there is a more than ±5 °C difference between this temperature and the temperature when the last auto calibration was run, the instrument generates an event in the event log and a warning message.

The auto calibration constants are stored through a power cycle. You do not need to run auto calibration because power has been cycled.

Auto calibration monitors the front-panel inputs of the instrument. If more than 30 V DC or 1 V AC is detected on the front-panel inputs, auto calibration is not run and an event message is displayed.

Auto calibration also monitors the temperature at the start and end of auto calibration. If the start and end temperature differs by more than ±1 °C, the auto calibration values are not stored and a warning message is generated.

Running auto calibration

You can run auto calibration as needed by using the front panel or remote command. If the instrument has completed the warm up period, auto calibration takes about six minutes to run. During auto calibration, you cannot use the instrument. A status message is displayed on the front panel of the instrument while auto calibration is running. At completion, a status message is generated.

> ⚠ CAUTION
>
> To prevent instrument damage, verify that no test voltages are connected to the input terminals when performing auto calibration.
>
> Do not cycle power during the auto calibration routine. Doing so could affect the accuracy of the instrument.

To prepare for auto calibration:

1. Disconnect any test cables from the input terminals.
2. Place the Model DMM7510 in a temperature-stable location.
3. Turn on instrument power and allow the instrument to warm up for at least 90 minutes. When the instrument has completed the warm-up period, a message is displayed and an information event is generated in the event log.

To run auto calibration from the front panel:

1. Press the **MENU** key.
2. Under System, select **Calibration**.
3. Select **Start ACAL**. A prompt is displayed.
4. Select **Yes**. A progress bar is displayed while the calibration runs.

To run auto calibration using SCPI commands:

Send:

```
ACAL:RUN
```

To run auto calibration using TSP commands:

Send:

```
acal.run()
```

NOTE

Once auto calibration has started, you cannot stop it. After completion, however, you can use remote commands to revert to the previous auto calibration settings. If you are using SCPI, refer to :ACAL:REVert (on page 6-17) for detail. If you are using TSP, refer to acal.revert() (on page 8-12).

Scheduling auto calibration

You can set up your instrument to run auto calibration automatically. You can also set up the instrument to prompt you to run auto calibration at regular intervals. To determine the best schedule for your application, see the Model DMM7510 specifications for detail on the accuracy with and without auto calibration.

Auto calibration does not start until actions that are active on the instrument are complete. For example, if a trigger model is running when auto calibration is scheduled to run, auto calibration does not start until the trigger model stops.

If the scheduled time for auto calibration occurs before the warmup period completes, the instrument will not start auto calibration. The instrument waits until the warmup period is complete before starting a scheduled auto calibration. A message is displayed when warmup is complete and auto calibration is going to run.

If the instrument is powered off when an auto calibration was scheduled, auto calibration is run as soon as the warmup period is complete when the instrument is powered on.

You can run auto calibration manually even if a scheduled auto calibration is set.

When auto calibration is scheduled to run at a scheduled interval, but it runs at a time other than the scheduled interval, subsequent scheduled intervals are adjusted according to the actual auto calibration start time.

From the front panel:

1. Press the **MENU** key.
2. Under System, select **Calibration**.
3. Select **Scheduling Action**. To have the instrument:
    -   Prompt you to run auto calibration: Select **Notify**.
    -   Run auto calibration at a specific time: Select **Run**.
    -   To stop scheduling: Select **None**. If you select None, you do not need to make additional settings.
4. Select **Scheduling Interval**.
5. Select the interval.
6. Select **Scheduled Time** to select the time when the auto calibration will run or when you will be prompted to run it.

To review the next schedule time and date, see the information listed next to Next Run.

Using SCPI commands:

Refer to :ACAL:SCHedule (on page 6-19).

Using TSP commands:

Refer to acal.schedule() (on page 8-13).

Reviewing calibration information

The Calibration screen displays information about the last auto calibration and factory calibrations that were run and the present status. For detail on this screen, refer to System Calibration menu (on page 2-50).

For auto calibration, you can also access this information from the commands in the SCPI ACAL subsystem (on page 6-13) or the TSP `acal.*` commands.

Monitoring internal temperature

You can monitor the temperature difference between the actual internal temperature and the temperature when auto calibration ran through the panel or by using remote commands. With remote commands, you can also check the present internal temperature and the internal temperature when auto calibration was last run. Temperature is returned in Celsius (°C).

The internal temperature is not updated on the Calibration screen until the warmup period is complete. The remote commands always return the present temperature.

From the front panel:
1. Press the **MENU** key.
2. Under System, select **Calibration**.
3. The Temperature Difference is displayed.

Using SCPI commands:

For the present internal temperature, send:

```
:SYSTem:TEMPerature:INTernal?
```

For the temperature difference, send:

```
:ACAL:LASTrun:TEMPerature:DIFFerence?
```

For the temperature when auto calibration was last run, send:

```
:ACAL:LASTrun:TEMPerature:INTernal?
```

Using TSP commands:

For the present internal temperature, send:

```
print(localnode.internaltemp)
```

For the temperature difference, send:

```
print(acal.lastrun.tempdiff)
```

For the temperature when auto calibration was last run, send:

```
print(acal.lastrun.internaltemp)
```

Digital I/O

The Model DMM7510 digital I/O port provides six independently configurable digital input/output lines.

You can use these lines for digital control by writing a bit pattern to the digital I/O lines. Digital control is used for applications such as providing binning codes to a component handler. Digital control uses the state of the line to determine the action to take.

You can also use these lines for triggering by using the transition of the line state to initiate an action. The instrument can generate output trigger pulses and detect input trigger pulses. Triggering is used for applications such as synchronizing the operations of a measure instrument with the operations of other instruments.

You cannot configure or directly control the digital I/O lines from the front panel. To configure and control any of the six digital input/output lines, you need to send commands to the Model DMM7510 over a remote interface. You can use either the SCPI or TSP command set. See Remote communications interfaces (on page 2-61) for information about setting up a remote interface and choosing a command set.

Digital I/O connector and pinouts

The digital I/O port uses a standard female DB-9 connector, which is located on the rear panel of the Model DMM7510. You can connect to the Model DMM7510 digital I/O using a standard male DB-9 connector. The port provides a connection point to each of the six digital I/O lines and other connections as shown in the following table.

Figure 115: Model DMM7510 digital I/O port
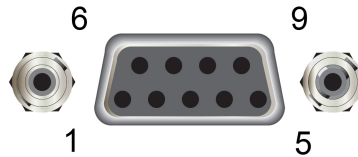
Model DMM7510 digital I/O port pinouts

Pin	Description
1	I/O line #1
2	I/O line #2
3	I/O line #3
4	I/O line #4
5	V_{ext} line (relay flyback diode protection; maximum 33 V)
6	I/O line #5
7	+5 V line. Use this pin to drive external logic circuitry. Maximum current output is 500 mA. This line is protected by a self-resetting fuse (one-hour recovery time).
8	I/O line #6
9	Ground

Digital I/O port configuration

The following figure shows the basic configuration of the digital I/O port.

To set a line high (nominally +5 V), write a `1` to it; to set a line low (nominally 0 V), write a `0` to it. To allow an external device to control the state of the line, the line must be set to input mode or open-drain mode. An attached device must be able to sink at least 50 μA from each I/O line.

Figure 116: Digital I/O port configuration
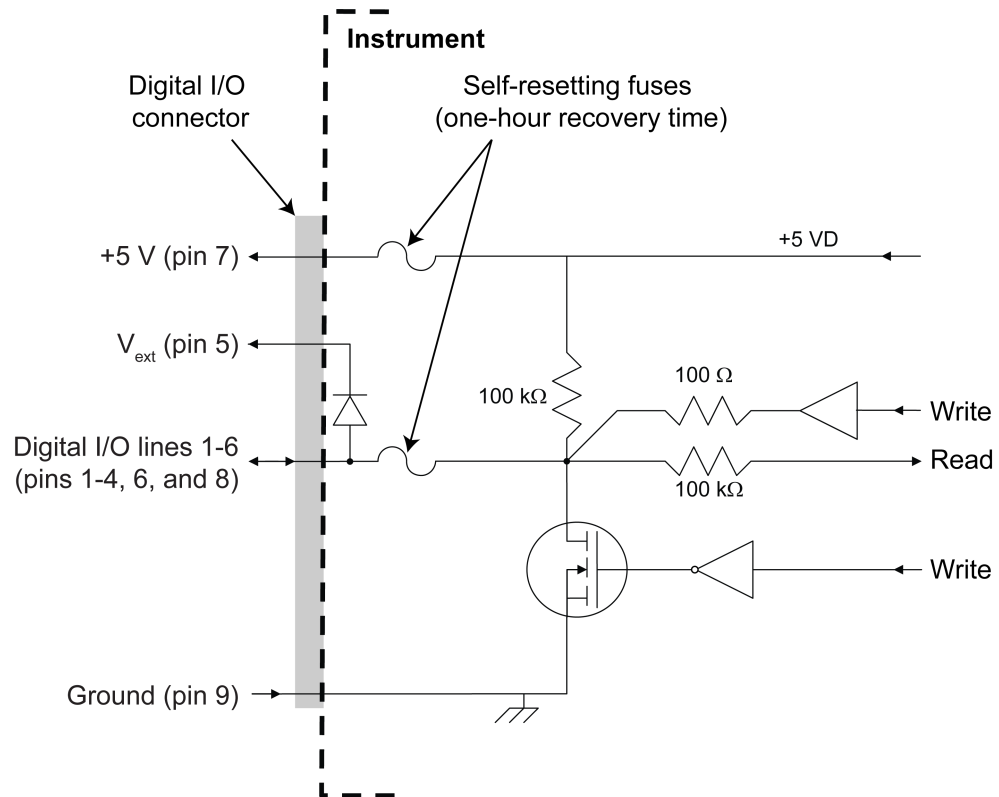
NOTE

For additional details about the digital output, see the Model DMM7510 specifications (available at the Keithley Instruments support website (*http://www.keithley.com/support*)).

Vext line

The digital I/O port provides a line (V_{ext}) with a flyback diode clamp that you can use when controlling inductive circuitry such as relay coils or low-power solenoids. You can use the built-in 5 V supply or an external voltage supply for these types of applications. The externally supplied voltage can be up to +33 V.

> ⚠ CAUTION
>
> Do not apply more than 50 mA (maximum current) or exceed +33 V (maximum voltage) on the digital I/O lines. Applying current or voltage exceeding these limits may damage the instrument.

Refer to the following figure for a simplified schematic of a sample control circuit for a relay. You can externally power a different device by replacing the relay coil with the other device. The relay is actuated by configuring the corresponding digital output line. Most of these types of applications use an active-low (set the bit to 0) to turn the relay on (ON = 0 V). In the low state (0 V), the output transistor sinks current through the external device. In the high state, the output transistor is off (transistor switch is open). This interrupts current flow through the external device.

Figure 117: Digital I/O Vext (example external circuit)
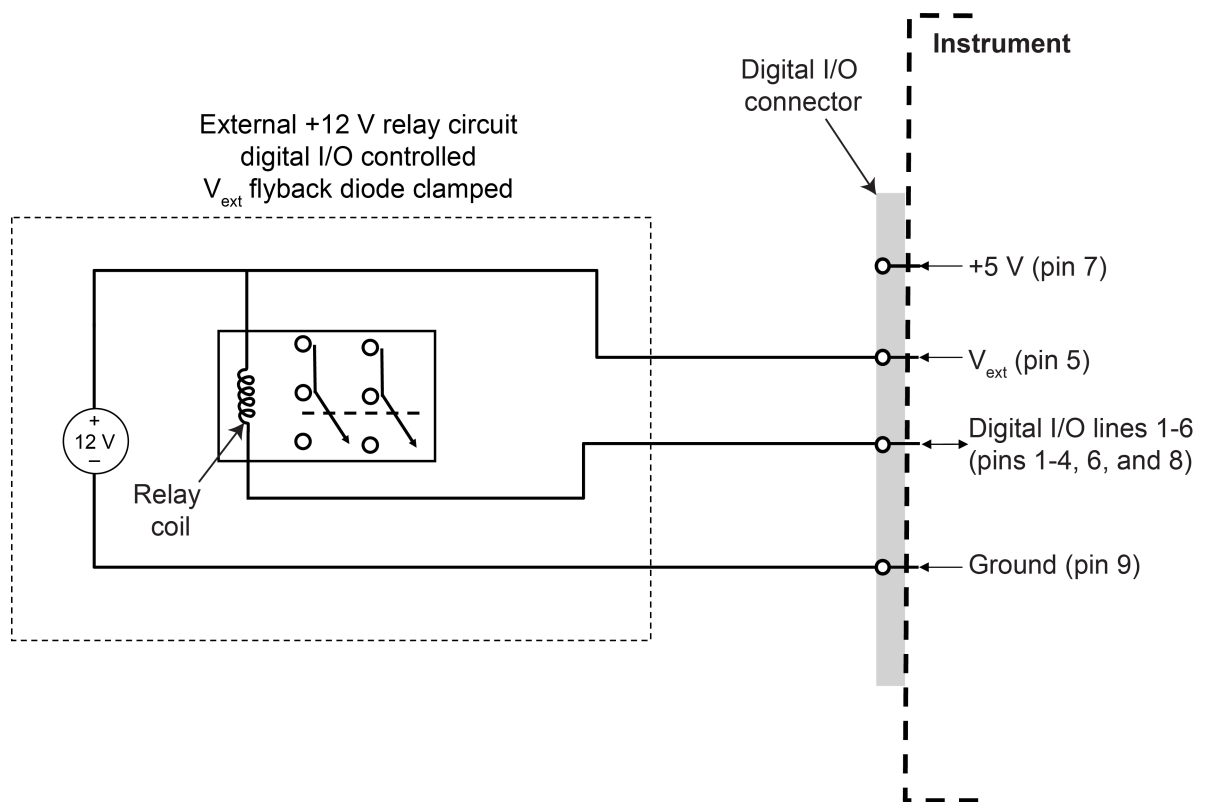
+5 V line

The digital I/O port provides a +5 V output. You can use this line to drive external circuitry. The maximum current output for this line is 500 mA. A self-resetting fuse with a one-hour recovery time protects this line.

If you are using this supply to drive a relay, it should be connected to the V_{ext} line so that the relay is protected by the flyback diode clamp.

Digital I/O lines

You can 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 acceptor

<div>

NOTE

When you configure the digital I/O lines for triggering applications, configure the output lines before the input lines. This prevents possible false input trigger detection in certain situations.

</div>

Digital control modes

If you are setting a line for digital control, you can set the line to be open-drain, output, or input, as described in the following topics.

Open-drain

When you place a line in open-drain mode, the line is configured to be an open-drain signal with a 100 kΩ pull-up resistor. This makes the line compatible with other instruments that use open-drain digital I/O lines, such as other Keithley Instruments products that only support open-drain for its digital I/O. In this mode, the line can serve as an input, an output, or both. You can read from the line or write to it. When a digital I/O line is used as an input in open-drain mode, you must write a `1` to the line to enable it to detect logic levels that are generated from external sources.

Output

When you place a line in output mode, 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. Unlike the input or open-drain modes, it will not respond to externally generated logic levels.

When you read the line, it shows the present output status and an event message is generated.

Input

The input mode is similar to the open-drain mode, except that a line in this mode is intended to be used strictly as an input. When you place a line in input mode, the instrument automatically writes a `1` to the line to enable it to detect externally generated logic levels.

You can read an input line, but you cannot write to it. You also cannot change the logic level while the line is in input mode. If you attempt to change the logic level of a line that is in input mode, an event message is generated.

Trigger control modes

You can use the trigger control modes to synchronize instrument operation with the operation of other instruments. These modes either detect or generate transitions in the state of the line, from high to low (falling edge) or from low to high (rising edge). The input edge detection setting of the Model DMM7510 determines which type of transition is detected as an input trigger. Output triggers are typically generated in the form of a pulse. The type of transition that occurs on the leading edge of the pulse is determined by an output logic setting. The duration of the pulse is determined by a pulse width setting.

You can use the trigger control modes with interactive triggering or with the trigger model. For more information about the trigger modes and triggering, refer to

Open-drain

When you set the instrument to trigger mode and place a line in open-drain mode, the line is configured to be an open-drain signal with a 100 kΩ pull-up resistor. This makes the line compatible with other instruments that use open-drain trigger signals, such as other Keithley Instruments products that only support open-drain for its digital I/O. In this mode, you can use the line to detect input triggers or generate output triggers, or both. To use this mode successfully, you must carefully configure the input edge and output logic settings because both of these affect the initial state of the trigger line. It is recommended that you reset the line before selecting and configuring this mode.

To use the line only as a trigger input:
1. Reset the line.
2. Set the input trigger edge detection type to falling, rising, or either.

The command that sets the detection type automatically sets the line high. This enables the line to respond to and detect externally generated triggers.

Do not set the output trigger logic type to positive after setting the edge detection type. This sets the line low, which will prevent the line from operating correctly as a trigger input.

To use the line only as a trigger output:
1. Reset the line.
2. Set the output trigger logic type to negative (falling edge) or positive (rising edge).

When you set the logic type to negative, the instrument automatically sets the line high. Setting the logic type to positive automatically sets the line low.

Do not set the input trigger edge detection type after setting the positive logic type. This will set the line high, which will prevent the line from operating correctly as a trigger output.

To use the line as both a trigger input and a trigger output (falling edge triggers only):
1. Reset the line.
2. Set the output trigger logic type to negative (falling edge).
3. Set the input trigger edge detection type to falling, rising, or either.

You can use these settings for triggering applications that use Keithley Instrument products that feature Trigger Link.

Output

When you place a line in output mode, it is automatically set high or low depending on the output logic setting. Use the negative logic setting when you want to generate a falling edge trigger. Use the positive logic setting when you want to generate a rising edge trigger. You cannot detect incoming triggers on a line configured as a trigger output.

Input

When you place a line in input mode, it is automatically set high to allow it to respond to and detect externally generated triggers. Depending on the input edge detection setting, the line can detect falling-edge triggers, rising-edge triggers, or both.

The line cannot generate an output trigger if it is set to the trigger input mode.

Synchronous triggering

The synchronous triggering modes allow you to:

- Implement bidirectional triggering on a single trigger line
- Start operations on one or more external instruments using a single trigger line
- Wait for all instruments to complete all triggered actions

To coordinate non-Keithley instrumentation with synchronous triggering, the non-Keithley instrument must have a trigger mode that is similar to the synchronous acceptor or synchronous master trigger mode.

To use synchronous triggering, configure the triggering master to synchronous master trigger mode or the non-Keithley equivalent. Configure all other instruments in the test system to the synchronous acceptor trigger mode or equivalent.

Synchronous master

Use the synchronous master trigger mode with the synchronous acceptor mode or its non-Keithley equivalent.

Configure only one instrument as a synchronous master. Configure all other instruments that are connected to the synchronization line as synchronous acceptors.

When a digital I/O line is set to the synchronous master mode, it generates falling edge output triggers and detects rising edge input triggers on the same trigger line.

Instruments that are configured as synchronous acceptors detect the falling-edge trigger and begin their triggered actions. At the same time, they latch the line low and hold it in that state until their triggered actions complete. Each instrument configured as an acceptor releases the line upon completion of its triggered actions.

When all instruments have released the line, the line changes state and generates a rising edge trigger. This trigger is detected by the synchronous master, which then performs its next triggered action.

Synchronous master detail

Input characteristics:

- All rising edges are input triggers.
- When all external drives release the physical line, the rising edge is detected as an input trigger.
- A rising edge is not detected until all external drives release the line and the line floats high.

Figure 118: Synchronous master input trigger
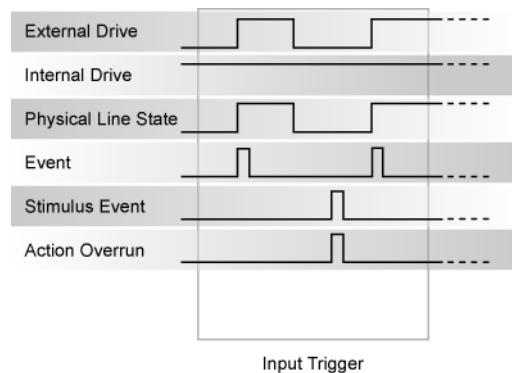
Output characteristics:

- In addition to trigger events from other trigger objects, the TSP commands `trigger.digout[N].assert()` and `trigger.tsplinkout[N].assert()` generate a low pulse that is similar to the falling-edge trigger mode.
- An action overrun occurs if the physical line state is low when a stimulus event occurs.

Figure 119: Synchronous master output trigger
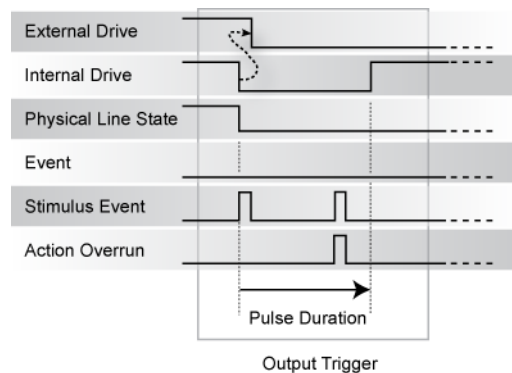
Synchronous acceptor

Use the synchronous acceptor trigger mode with the synchronous master mode or its non-Keithley equivalent.

Only one instrument should be configured as a synchronous master. All other instruments connected to the synchronization line must be configured as synchronous acceptor or equivalent.

A line that is set to the synchronous acceptor mode detects falling edge input triggers and generates rising edge output triggers on the same trigger line. When a line that is configured as synchronous acceptor detects the falling edge trigger, it latches the line low and holds it in that state until all triggered actions for that instrument are complete. When the triggered actions are complete, the synchronous acceptor line releases the line. When all connected instruments have released the line, the line changes state and generates a rising edge trigger.

Synchronous acceptor detail

Input characteristics:

- The falling edge is detected as the external drive pulses the line low, and the internal drive latches the line low.

Figure 120: Synchronous acceptor input trigger
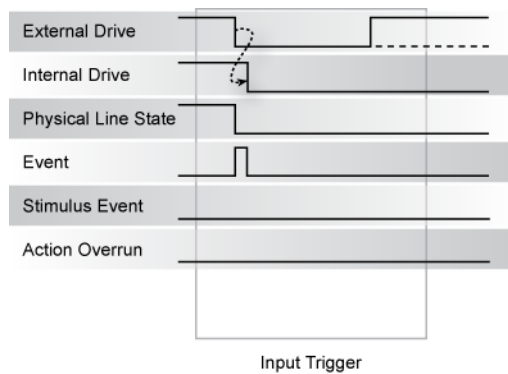
Output characteristics:

- In addition to trigger events from other trigger objects, the TSP commands `trigger.digout[N].assert()` and `trigger.tsplinkout[N].assert()`
- The physical line state does not change until all drives (internal and external) release the line.
- Action overruns occur if the internal drive is not latched low and a source event is received.

Figure 121: Synchronous acceptor output trigger
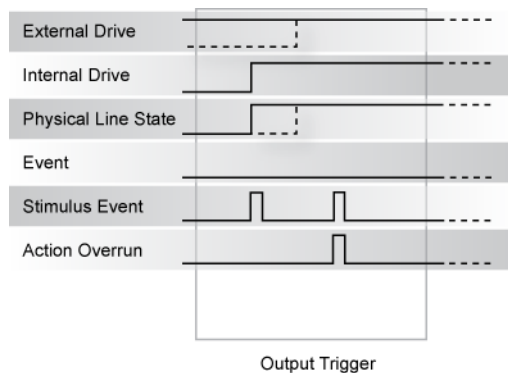
Remote digital I/O commands

Summaries of SCPI and TSP comands are in the following table. You can use the digital I/O port to:

- Perform basic steady-state digital I/O operations, such as reading and writing to individual I/O lines or reading and writing to the entire port
- Trigger the Model DMM7510 when external trigger pulses are applied to the digital I/O port
- Provide trigger pulses to external devices

SCPI command TSP command	Description
:DIGital:LINE<n>:MODE (on page 6-34) digio.line[N].mode (on page 8-48)	This command sets the mode of the digital I/O line to be a digital line, trigger line, or synchronous line and sets the line to be input, output, or open-drain.
A line reset is not available in SCPI; however, the line is reset when a global reset (*RST) is sent digio.line[N].reset() (on page 8-50)	This command resets digital I/O line values to their factory defaults.
:DIGital:LINE<n>:STATe (on page 6-36) digio.line[N].state (on page 8-51)	This command sets a digital I/O line high or low when the line is set for digital control and returns the states on the digital I/O lines.
:DIGital:READ? (on page 6-37) digio.readport() (on page 8-51)	This command reads the digital I/O port. All six lines must be configured as digital control lines. If not, this command generates an error.
:DIGital:WRITe <n> (on page 6-38) digio.writeport() (on page 8-52)	This command writes to all digital I/O lines. All six lines must be configured as digital control lines. If not, this command generates an error.
:TRIGger:DIGital<n>:IN:CLEar (on page 6-194) trigger.digin[N].clear() (on page 8-257)	This command clears the trigger event on a digital input line.
:TRIGger:DIGital<n>:IN:EDGE (on page 6-194) trigger.digin[N].edge (on page 8-257)	This command sets the edge used by the trigger event detector on the given trigger line.
:TRIGger:DIGital<n>:IN:OVERrun? (on page 6-196) trigger.digin[N].overrun (on page 8-258)	This command returns the event detector overrun status.
Not available in SCPI trigger.digin[N].wait() (on page 8-259)	This command waits for a trigger.
Not available in SCPI trigger.digout[N].assert() (on page 8-260)	This command asserts a trigger pulse on one of the digital I/O lines.
:TRIGger:DIGital<n>:OUT:LOGic (on page 6-196) trigger.digout[N].logic (on page 8-261)	This command sets the output logic of the trigger event generator to positive or negative for the specified line.
:TRIGger:DIGital<n>:OUT:PULSewidth (on page 6-197) trigger.digout[N].pulsewidth (on page 8-261)	This command describes the length of time that the trigger line is asserted for output triggers.
Not available in SCPI trigger.digout[N].release() (on page 8-262)	This command releases an indefinite length or latched trigger.
:TRIGger:DIGital<n>:OUT:STIMulus (on page 6-198) trigger.digout[N].stimulus (on page 8-263)	This command selects the event that causes a trigger to be asserted on the digital output line.

NOTE

To use the trigger model as a stimulus to a digital I/O line, you can use the trigger model Notify block. For information on the Notify block, see Notify building block (on page 3-80).

Digital I/O bit weighting

Bit weighting for the digital I/O lines is shown in the following table. Line 1 is the least significant bit.

Line #	Bit	Pin	Decimal	Hexadecimal	Binary
1	B1	1	1	0x01	000001
2	B2	2	2	0x02	000010
3	B3	3	4	0x04	000100
4	B4	4	8	0x08	001000
5	B5	6	16	0x10	010000
6	B6	8	32	0x20	100000

Digital I/O programming examples

These examples provide typical methods you can use to work with the digital I/O port.

Outputting a bit pattern

The programming examples below illustrate how to output the bit pattern 110101 at the digital I/O port. Line 1 (bit 1) is the least significant bit.

Using SCPI commands to configure all six lines as digital outputs:

```
:DIGital:LINE1:MODE DIGital, OUT
:DIGital:LINE2:MODE DIGital, OUT
:DIGital:LINE3:MODE DIGital, OUT
:DIGital:LINE4:MODE DIGital, OUT
:DIGital:LINE5:MODE DIGital, OUT
:DIGital:LINE6:MODE DIGital, OUT
```

Using SCPI commands to set the state of each line individually:

```
:DIGital:LINE6:STATe 1
:DIGital:LINE5:STATe 1
:DIGital:LINE4:STATe 0
:DIGital:LINE3:STATe 1
:DIGital:LINE2:STATe 0
:DIGital:LINE1:STATe 1
```

Using SCPI commands to set all six lines at once by writing to the port the decimal equivalent of the bit pattern:

```
:DIGital:WRITe 53
```

Using TSP commands to configure all six lines as digital outputs:

```
-- Send for loop as a single chunk or include in a script.
for i = 1, 6 do
    digio.line[i].mode = digio.MODE_DIGITAL_OUT
end
```

Using TSP commands to set the state of each line individually:

```
digio.line[1].state = digio.STATE_HIGH
digio.line[2].state = digio.STATE_LOW
digio.line[3].state = digio.STATE_HIGH
-- You can use 0 instead of digio.STATE_LOW.
digio.line[4].state = 0
-- You can use 1 instead of digio.STATE_HIGH.
digio.line[5].state = 1
digio.line[6].state = 1
```

Using TSP commands to set all six lines at once by writing to the port the decimal equivalent of the bit pattern:

```
-- You can write binary, decimal or hexadecimal values, as shown below.
-- Use binary value.
digio.writeport(0b110101)
-- Use decimal value.
digio.writeport(53)
-- Use hexadecimal value.
digio.writeport(0x35)
```

Reading a bit pattern

The programming examples below illustrate how to read part or all of a bit pattern that has been applied to the digital I/O port by an external instrument. Assume the binary pattern is 100101. Line 1 (bit 1) is the least significant bit.

Using SCPI commands:

Configure all 6 lines as digital inputs:

```
DIGital:LINE1:MODE DIGital, IN
DIGital:LINE2:MODE DIGital, IN
DIGital:LINE3:MODE DIGital, IN
DIGital:LINE4:MODE DIGital, IN
DIGital:LINE5:MODE DIGital, IN
DIGital:LINE6:MODE DIGital, IN
```

Read the state of Line 2:

```
DIGital:LINE2:STATe?
```

Value returned is 0.

Read the state of Line 3:

```
DIGital:LINE3:STATe?
```

Value returned is 1.

Read the value applied to the entire port:

```
DIGital:READ?
```

Value returned is 37, which is the decimal equivalent of the binary bit pattern.

Using TSP commands:

```
-- Configure all six digital I/O lines as digital inputs.
-- You can also use a for loop.
digio.line[1].mode = digio.MODE_DIGITAL_IN
digio.line[2].mode = digio.MODE_DIGITAL_IN
digio.line[3].mode = digio.MODE_DIGITAL_IN
digio.line[4].mode = digio.MODE_DIGITAL_IN
digio.line[5].mode = digio.MODE_DIGITAL_IN
digio.line[6].mode = digio.MODE_DIGITAL_IN
-- Read and then print the state of Line 2 (bit 2).
b2 = digio.line[2].state
print(b2)
```

The value returned is `digio.STATE_LOW`.

```
-- Print the state of Line 3 (bit 3).
print(digio.line[3].state)
```

The value returned is `digio.STATE_HIGH`.

```
-- Read and then print the value applied to the entire port.
port = digio.readport()
print(port)
```

The value returned is `37`, which is the decimal equivalent of the binary bit pattern.

External I/O

The external I/O (EXT TRIG IN/OUT) connector on the rear panel of the instrument is a TTL-compatible input/output line with a 0 to 5 V logic signal. You can use this line for triggering by using the transition of the line state to initiate an action. The instrument can generate output trigger pulses and detect input trigger pulses on this line.

Triggering is used for applications such as synchronizing the operations of the Model DMM7510 with the operations of other instruments.

Setting up the external I/O

You cannot configure or directly control the external I/O lines from the front panel. To configure and control the external I/O line, you need to send commands to the Model DMM7510 over a remote interface. You can use either the SCPI or TSP command set. See Remote communications interfaces (on page 2-61) for information about setting up a remote interface and choosing a command set.

The options to set up the external I/O are stimulus, edge, and logic.

The stimulus selects the event that causes a trigger to be asserted on the external output line. You can use any of the standard trigger events with the external I/O. See Trigger events (on page 3-97) for a list of the available trigger events.

The edge sets the type of edge that is detected as an input on the external in line. You can set the external I/O to detect trigger inputs on the falling edge, rising edge, or either edge. When falling edge is selected, the input is detected when the line state transitions from high to low. When rising edge is selected, the input is detected when the line state transitions from low to high.

The logic type determines if the external I/O output asserts a TTL-high pulse or a TTL-low pulse for the trigger.

You can use the external I/O with interactive triggering or with the trigger model. For more information about the trigger modes and triggering, refer to Triggering (on page 3-59).

External trigger I/O pinouts

The EXT TRIG IN/OUT pinouts are shown in the following figure and table.

Figure 122: EXT TRIG I/O pinout diagram

Pin 2   Pin 1
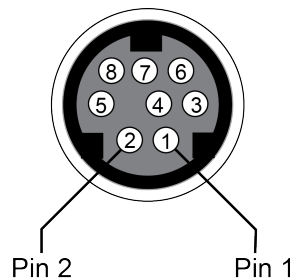
Pin	Description
1	EXT TRIG OUT
2	EXT TRIG IN
3	Unused
4	Unused
5	Unused
6	Unused
7	Signal ground
8	Signal ground

The EXT TRIG IN can detect the falling edge, rising edge, or either edge pulse. The detected edge is set through edge settings using remote commands. The default is falling edge. The following figure shows the electrical and timing specifications for pulse detection of EXT TRIG IN.

Figure 123: External I/O pulse specifications
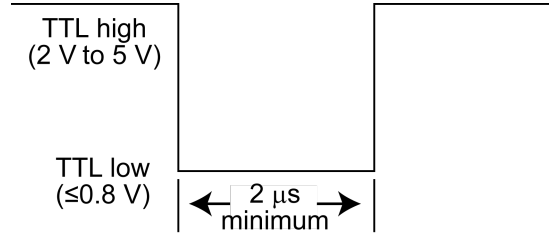
Remote external I/O commands

Commands for both SCPI and TSP are summarized in the following table.

SCPI command TSP command	Description
A line reset is not available in SCPI; however, the line is reset when a global reset (*RST) is sent trigger.ext.reset() (on page 8-264)	This command resets the edge, logic, and stimulus values for the external in/out port to their default values.
:TRIGger:EXTernal:IN:CLEar (on page 6-199) trigger.extin.clear() (on page 8-265)	This command clears the trigger event on the external in line.
:TRIGger:EXTernal:IN:EDGE (on page 6-199) trigger.extin.edge (on page 8-265)	This command sets the type of edge that is detected as an input on the external in line.
:TRIGger:EXTernal:IN:OVERrun? (on page 6-200) trigger.extin.overrun (on page 8-266)	This command returns the event detector overrun status.
Not available in SCPI trigger.extin.wait() (on page 8-266)	This command waits for an input trigger.
Not available in SCPI trigger.extout.assert() (on page 8-267)	This command asserts a trigger on the external I/O line.
:TRIGger:EXTernal:OUT:LOGic (on page 6-201) trigger.extout.logic (on page 8-268)	This command sets the output logic of the trigger event generator to positive or negative for the external out line.
:TRIGger:EXTernal:OUT:STIMulus (on page 6-202) trigger.extout.stimulus (on page 8-268)	This command selects the event that causes a trigger to be asserted on the external output line.

NOTE

To use the trigger model as a stimulus to the external I/O line, you can use the trigger model Notify block. For information on the Notify block, see Notify building block (on page 3-80).

Measurement methods

Triggers are signals that instruct the instrument to make a measurement. You can set the Model DMM7510 to use the following triggering measurement methods:

- Continuous measurement: The instrument is making measurements continuously.
- Manual trigger mode: Press the front-panel TRIGGER key to initiate a single measurement.
- Trigger model: The instrument makes measurements according to the settings of the trigger model. To select this method, a trigger model must be set up. Select Initiate Trigger Model to start the trigger model, or Abort Trigger Model to stop a trigger model that is presently running.

Continuous measurement triggering

When you select the continuous measurement method, the instrument makes measurements continuously.

The continuous measurement method is only available when you are controlling the instrument locally (through the front panel).

The instrument stores the readings in the active reading buffer. See Reading buffers (on page 3-12) for detail on the buffer options that are available.

If you press the front-panel **TRIGGER** key when the instrument is set to the continuous measurement method, measurements are not made. Instead, a dialog box is displayed that asks if you want to change the measurement method.

Trigger key triggering

When you select the manual trigger key method of triggering, the instrument only makes a measurement when you press the front-panel **TRIGGER** key.

The manual trigger key measurement method is only available when you are controlling the instrument through the front panel.

The instrument stores the readings in the active reading buffer. See Reading buffers (on page 3-12) for detail on the buffer options that are available.

Trigger model triggering

When you select the trigger model measurement method, the instrument uses a trigger model to control the sequence in which measurements occur. The Model DMM7510 trigger model is flexible, allowing you to control as much or as little as needed for your measurement application.

When you are remotely controlling the instrument, the trigger model measure method is automatically selected. In addition, you can view different buffers from the front panel, but actual buffer that is used is defined by the remote commands.

For detail on the trigger model, see Trigger model (on page 3-72).

Switching between measurement methods

The measurement methods that are available to you depend on how you are controlling the instrument.

If you are using the front panel to control the instrument, you can choose any of the measurement methods.

If you are using a remote interface to control the instrument, you can only use the trigger model measurement method. When you switch to a remote interface, the trigger model measurement method is automatically selected. If you switch from remote control to front-panel control, the trigger model measurement method remains selected.

If you are running a script, the instrument automatically switches to the trigger model measurement method.

Using the front panel:
1. Press the front-panel **TRIGGER** key for 2 s. A dialog box displays the available trigger methods. The presently selected method is highlighted.
2. Select the method you want to use.
3. If the instrument is in remote control, the instrument displays a confirmation dialog box. Select **Yes** to change to local control.

Triggering

Triggering allows you to start and synchronize measure actions on one or more instruments with a trigger event or a combination of trigger events that you set. This section describes some of the options available for triggering, including command interface triggering, timers, analog trigger, and event blenders.

Command interface triggering

A command interface trigger event occurs when:

- A GPIB GET command is detected (GPIB only)

- A VXI-11 `device_trigger` method is invoked (VXI-11 only)

- A `*TRG` message is received

To use a command interface trigger event as an input stimulus for another trigger object, set the stimulus as TSP event `trigger.EVENT_COMMAND` or the SCPI event `COMMAND`. To ensure that trigger commands that are issued over the command interface are processed in the correct order, the instrument does not generate a trigger event until:

- The trigger command is executed

- TSP only: `trigger.wait()` retrieves the trigger command from the command queue before it would normally be executed

Command interface triggering does not generate action overruns. The triggers are processed in the order that they are received in the Model DMM7510 command queue. The Model DMM7510 only processes incoming commands when no commands are running. Unprocessed input triggers can cause an overflow in the command queue. It is important to make sure a script processes triggers while it is running.

> ## NOTE
>
> If you are using a test script using TSP, the command queue can fill up with trigger entries if over 50 `*TRG` messages are received while a test script is running, even if the script is processing triggers. You can avoid this by using the localnode.prompts4882 (on page 8-221) attribute, and by using `trigger.wait()` calls that remove the `*TRG` messages from the command queue. If the command queue fills with too many trigger entries, messages such as `abort` are not processed.

Triggering using hardware lines

You can use the digital I/O lines, external I/O, and TSP-Link® synchronization lines to synchronize the operations of the Model DMM7510 with those of external instruments. You can use these lines to synchronize the Model DMM7510 with other TSP-enabled instruments, including other Model DMM7510 instruments. You must use the digital I/O lines or external I/O line to synchronize the Model DMM7510 with other Keithley products or other non-Keithley products.

The lines are configured and controlled similarly. See Digital I/O, TSP-Link System Expansion Interface (on page 3-102), and External I/O (on page 3-55) for more information about connections and configuration and control of the lines.

Analog triggering overview

You can use input signals for triggering when you are measuring current or voltage using the DC measure or digitize functions. The instrument compares the signals, before they are processed into measurements, to the settings that you define. The trigger occurs when the signal satisfies the specified conditions. Triggers generated by these comparisons are called analog triggers. You can use analog triggers to trigger instrument action in the same ways that you use other trigger types.

Analog triggers should be set after other instrument settings. In particular, if you are using a measure function, select the function, turn autozero off, and select a specific range before setting up analog triggers.

Analog trigger mode

To set up an analog trigger, you need to define the analog trigger mode. The mode defines how the instrument processes the signal that generates the trigger event. The available modes are edge, pulse, and window.

When edge is selected, the trigger event occurs when the signal crosses a level that you define. You also specify if the trigger occurs on the rising or falling edge of the signal.

The pulse and window modes are typically used to spot signal anomalies.

When pulse is selected, the trigger event occurs when a pulse satisfies the amplitude, polarity, and pulse width requirements that you specify.

When window is selected, the trigger event occurs when the signal enters or exits a window that is defined by low and high signal levels.

Edge mode

Edge triggers occur when you cross a defined signal level. When you select the edge trigger mode, you also need to set the trigger level and the slope (rising or falling). Refer to the following figure for an example signal.

Figure 124: Edge analog trigger mode
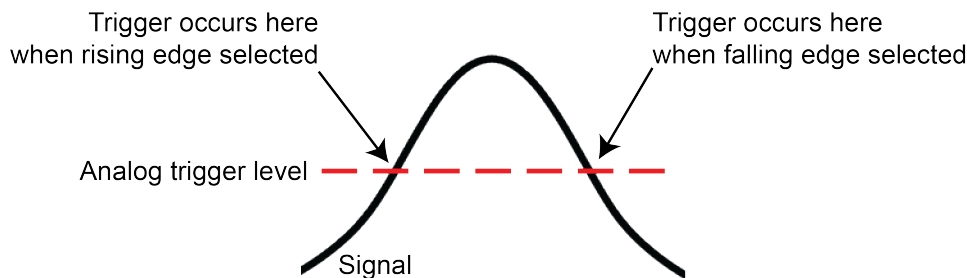
Pulse mode

Pulse triggers occur when two complementary signal edges cross the trigger level and meet the polarity and timing constraints that you specify.

When you set up pulse mode, you define the level, width, condition, and polarity.

The level defines the pulse level that generates a trigger event.

The width defines the threshold value for the pulse width.

The condition defines whether the incoming pulse must have a duration greater than or less than the threshold pulse width.

The polarity determines if the trigger should occur when the pulse occurs above or below the trigger level.

Figure 125: Pulse analog trigger mode
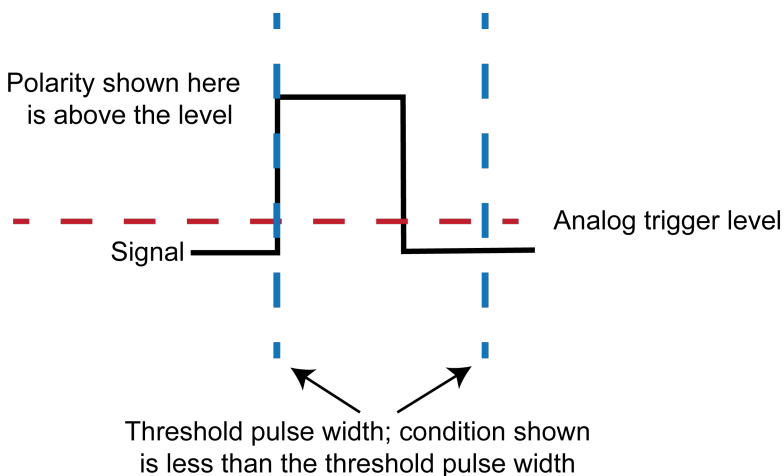
Window mode

Window triggers occur when a signal enters or leaves a defined signal window.

When you set up window mode, you define the high and low analog trigger levels. You also define whether the trigger should occur when the signal enters or leaves the window.

Figure 126: Window analog trigger mode
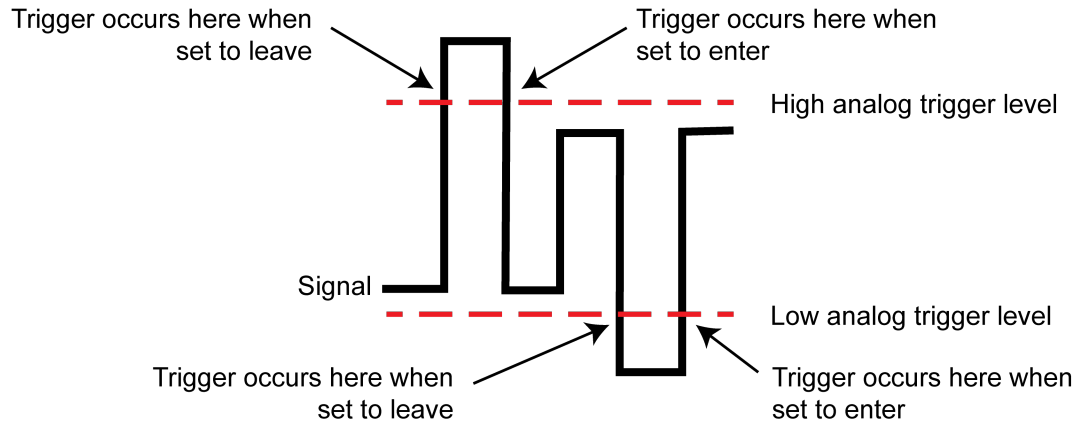
High frequency rejection

You can enable high frequency rejection for any analog trigger mode. False triggering around the set analog trigger level may occur with low frequency signals that are noisy, DC, or have low amplitude and slew rate during the peaks of input sine waves less than 250 Hz. High frequency rejection avoids the false triggers by the requiring the trigger event to be sustained for at least 60 μs. This behavior is similar to a low pass filter effect with a 4 kHz 3 dB bandwidth.

When high frequency rejection is on, 60 μs of additional trigger latency is incurred. You may also need to adjust the trigger levels to ensure that trigger condition is satisfied for at least 60 μs.

Analog triggering example with digitize function

This example shows how to set up analog trigger for a digitize function. The trigger is used to pulse the external out.

When the script runs, it opens the Graph tab on the front panel. To view the data, open the Scale tab and set X-Axis Method to Track Group.

Analog trigger example — SCPI

This code resets the instrument, and sets it to measure digitize volts on the 10 V range. Sample rate is set to 100,000 samples per second on the fastest aperture. DC coupling and input impedance are selected. The digitize count is set to make 1000 readings for each trigger condition.

The buffer size is set to 100,000 readings.

The analog trigger is set to edge with a rising slope and level of 0.5 V. High frequency rejection is disabled to allow triggering on a fast edge.

Configures the instrument to run the digitizer every time the analog trigger conditions are met and push external trigger pulses out that are synchronized with the analog trigger.

Opens the graph screen.

```
*RST
:SENSe:DIGitize:FUNCtion "VOLT"
:RANGe:DIGitize:VOLTage 10
:SENSe:DIGitize:VOLTage:SRATe 100000
:SENSe:DIGitize:VOLTage:APERture 1e-6
:SENSe:DIGitize:VOLTage:COUPling DC
:SENSe:DIGitize:VOLTage:INPutimpedance AUTO
:SENSe:DIGitize:COUNt 1000
:TRACe:POINts 100000, "defbuffer1"
:DIG:VOLT:ATR:MODE EDGE
:DIG:VOLT:ATR:EDGE:SLOP RIS
:DIG:VOLT:ATR:EDGE:LEV 0.5
:DIG:VOLT:ATR:HFR OFF
:SENSe:TRIGger:DIGitize:STIMulus ATRigger
:TRIGger:EXTernal:OUT:LOGic POSitive
:TRIGger:EXTernal:OUT:STIMulus ATRigger
:DISPlay:SCReen GRAPh
```

Analog trigger example — TSP

```
reset()
-- Select digitize volts function
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
-- Set the range to 10 V
dmm.digitize.range = 10
-- Set the sample rate to 100,000 samples per second
dmm.digitize.samplerate = 100000
-- Fastest aperture (set to 0 for AUTO)
dmm.digitize.aperture = 1e-6
-- Set coupling to DC to measure both AC and DC signal components
dmm.digitize.coupling.type = dmm.COUPLING_DC
-- Set input impedance to automatic
dmm.digitize.inputimpedance = dmm.IMPEDANCE_AUTO
-- Set count to 1000 readings per trigger condition
dmm.digitize.count = 1000
-- Set the buffer size to 100,000
defbuffer1.capacity = 100000
-- Set analog trigger mode to edge with a rising slope and level of 0.5 V
dmm.digitize.analogtrigger.mode = dmm.MODE_EDGE
dmm.digitize.analogtrigger.edge.slope = dmm.SLOPE_RISING
dmm.digitize.analogtrigger.edge.level = 0.5

-- Disable high frequency rejection to allow triggering on a fast edge
dmm.digitize.analogtrigger.highfreqreject = dmm.OFF

-- Digitize readings every time the analog trigger conditions are met
dmm.trigger.digitize.stimulus = trigger.EVENT_ANALOGTRIGGER

-- Push external trigger pulses out synchronized with the analog trigger
trigger.ext.reset()
trigger.extout.logic = trigger.LOGIC_POSITIVE
trigger.extout.stimulus = trigger.EVENT_ANALOGTRIGGER

-- Alternate: Push external trigger pulses out an I/O synced with the analog
   trigger
-- digio.line[1].mode = digio.MODE_TRIGGER_OUT
-- trigger.digout[1].logic = trigger.LOGIC_NEGATIVE
-- trigger.digout[1].stimulus = trigger.EVENT_ANALOGTRIGGER
-- trigger.digout[1].pulsewidth = 10e-6

-- Open the graph screen
display.changescreen(display.SCREEN_GRAPH)
```

LAN triggering overview

You can send and receive triggers over the LAN interface. The Model DMM7510 supports LAN extensions for instrumentation (LXI). It has eight LAN triggers that generate and respond to LXI trigger packets.

Understanding hardware value and pseudo line state

LAN triggering and hardware synchronization are similar, except that LAN triggering uses LXI trigger packets instead of hardware signals. A bit in the LXI trigger packet called the hardware value simulates the state of a hardware trigger line. The Model DMM7510 stores the hardware value as the pseudo-line state. Only the state of the last LXI trigger packet that was sent or received is stored.

The stateless event flag is a bit in the LXI trigger packet that indicates if the hardware value should be ignored. If it is set, the Model DMM7510 ignores the hardware value of the packet and generates a trigger event. The Model DMM7510 always sets the stateless flag for outgoing LXI trigger packets. If the stateless event flag is not set, the hardware value indicates the state of the signal.

The instrument interprets changes in the hardware value of consecutive LXI trigger packets as edge transitions. Edge transitions generate trigger events. If the hardware value does not change between successive LXI trigger packets, the Model DMM7510 assumes an edge transition was missed and generates a trigger event. The following table shows edge detection in LAN triggering.

LXI trigger edge detection

Stateless event flag	Hardware value	Pseudo-line state	Falling edge	Rising edge
0	0	0	Detected	Detected
0	1	0	-	Detected
0	0	1	Detected	-
0	1	1	Detected	Detected
1	-	-	Detected	Detected

You can set the LAN trigger edge detection method in incoming LXI trigger packets. The edge that is selected also determines the hardware value in outgoing LXI trigger packets. The following table lists the LAN trigger edges.

Trigger mode	Input detected	Output generated
Either edge	Either	Negative
Falling edge	Falling	Negative
Rising edge	Rising	Positive

LAN trigger objects generate LXI trigger events. LAN trigger objects generate LXI trigger events, which are LAN0 to LAN7 (zero based). To specify the LAN trigger event in a command, use `LANN`, where *N* is 1 to 8. `LAN1` corresponds to LXI trigger event LAN0 and `LAN8` corresponds to LXI trigger event LAN7. To specify the LAN trigger event in a command, see

Generate LXI trigger packets

You can configure the Model DMM7510 to output an LXI trigger packet to other LXI instruments.

To generate LXI trigger packets:

1. Call the SCPI `:TRIGger:LAN<n>:OUT:CONNect:STATe` command or TSP `trigger.lanout[N].connect()` function.

2. Select the event that triggers the outgoing LXI trigger packet by assigning the specific event to the LAN stimulus input using the SCPI `:TRIGger:LAN<n>:OUT:STIMulus` command or TSP `trigger.lanout[N].stimulus` attribute.

Make sure to use the same LXI domain on both the Model DMM7510 instrument and the other instrument. If the Model DMM7510 has a different LXI domain from the instrument at the other end of the trigger connection, the LXI trigger packets are ignored by both instruments.

Trigger timers

You can use trigger timers to add delays and start measurements at timed intervals. The Model DMM7510 has four independent timers that you can use.

Trigger timers are only available over a remote interface. You can set the count, delay, and when the trigger occurs for the trigger timers. You need to enable the trigger timers before using them using the SCPI `:TRIGger:TIMer<n>:STATe` or the TSP `trigger.timer[N].enable` command.

Count

The count sets the number of events to generate each time the timer generates a trigger event. Each event is separated by the delay set by the SCPI `:TRIGger:TIMer<n>:DELay` or TSP `trigger.timer[N].delay` command.

To configure the count, use the SCPI command `:TRIGger:TIMer<n>:COUNt` or the TSP command `trigger.timer[N].count`.

If *count* is set to a number greater than 1, the timer automatically starts the next trigger timer delay at the expiration of the previous delay.
Set *count* to zero (0) to cause the timer to generate trigger events indefinitely.

If you use the trigger timer with a trigger model, make sure the count value is the same or more than any count values expected in the trigger model.

Timer delays

You can set up the timers to perform delays. A delay is the period after the timer is triggered and before the timer generates a trigger event. All delay values are specified in seconds.

Delay lists, which are only available through TSP, allow the timer to sequence through an array of delay values. Delay lists allow the timer to use a different interval each time it performs a delay. Each time the timer generates a trigger event, it uses the next delay in the list. The timer repeats the delay list after all of the elements in the delay list have been used.

Using SCPI commands:

To set up a 50 µs trigger timer delay for timer 2, send the command:

```
TRIGger:TIMer2:DELay 50E-6
```

Using TSP commands to create a reading buffer:

To set up a 50 µs trigger timer delay for timer 2, send the command:

```
trigger.timer[2].delay = 50e-6
```

To set up a delay list for timer 3 for delays of 2, 10, 15, and 7 s, send the command:

```
trigger.timer[3].delaylist = {2, 10, 15, 7}
```

Define when to generate a timer event

You can specify when timer events are generated using the SCPI
`:TRIGger:TIMer<n>:STARt:GENerate` or TSP `trigger.timer[N].start.generate`
command.
When this is set to on, a trigger event is generated immediately when the timer is triggered.

When it is set to off, a trigger event is generated when the timer elapses.

You can also watch for a stimulus before starting the timer by using the SCPI
`:TRIGger:TIMer<n>:STARt:STIMulus` command or `trigger.timer[N].start.stimulus`
command.

You can also set an alarm or time in the future when the timer will start using the seconds and fractional seconds commands. (SCPI commands `:TRIGger:TIMer<n>:STARt:SEConds` and
`:TRIGger:TIMer<n>:STARt:FRACtional`; TSP commands
`trigger.timer[N].start.seconds` and `trigger.timer[N].start.fractionalseconds`.)

Timer action overruns

The timer generates an action overrun when it generates a trigger event while a timer delay is still in progress. Use the status model to monitor for the occurrence of action overruns. For details, see the Status model (on page C-1).

Using trigger timers with timing building blocks

For precise timing or if you need to synchronize timing with other execution blocks or events, you can use the SCPI or TSP trigger timer commands with trigger model wait blocks and notify blocks. You can use the trigger timer commands to add small precise delays or to start measurements or to overcome variable measurement delays. The Model DMM7510 has 1 to 4 independent timers.

For example, you can use a trigger timer to control the delay between non-sequential blocks. After creating a trigger timer, you can insert a notify block to start the timer at a specific point in the trigger model. You could then add a wait block to wait for the timer to expire.

Another example is a measure block that takes a variable amount of time. To ensure a precise time between measurements, you can create a trigger timer and define it to be a fixed interval that is longer than the longest possible measurement. Then you can set up the trigger model to include:

- A notify block that starts the trigger timer
- A measure block that makes a measurement
- A wait block that waits for the timer to expire
- A branch counter block that iterates some number of times

NOTE

Some attributes of trigger timers should not be used with the trigger model. Attributes you should not set are:

- Count value of 0 (resulting in generation of trigger events indefinitely)
- Delay lists

Remote trigger timer commands

SCPI trigger timer commands:

- :TRIGger:TIMer<n>:CLEar (on page 6-224)
- :TRIGger:TIMer<n>:COUNt (on page 6-224)
- :TRIGger:TIMer<n>:DELay (on page 6-226)
- :TRIGger:TIMer<n>:STARt:FRACtional (on page 6-227)
- :TRIGger:TIMer<n>:STARt:GENerate (on page 6-228)
- :TRIGger:TIMer<n>:STARt:OVERrun? (on page 6-228)
- :TRIGger:TIMer<n>:STARt:SEConds (on page 6-229)
- :TRIGger:TIMer<n>:STARt:STIMulus (on page 6-230)
- :TRIGger:TIMer<n>:STATe (on page 6-231)

TSP trigger timer commands:

- trigger.timer[N].clear() (on page 8-323)
- trigger.timer[N].count (on page 8-323)
- trigger.timer[N].delay (on page 8-325)
- trigger.timer[N].delaylist (on page 8-325)
- trigger.timer[N].enable (on page 8-327)
- trigger.timer[N].reset() (on page 8-328)
- trigger.timer[N].start.fractionalseconds (on page 8-329)
- trigger.timer[N].start.generate (on page 8-329)
- trigger.timer[N].start.overrun (on page 8-330)
- trigger.timer[N].start.seconds (on page 8-331)
- trigger.timer[N].start.stimulus (on page 8-331)
- trigger.timer[N].wait() (on page 8-332)

Event blenders

The ability to combine trigger events is called event blending. You can use an event blender to wait for up to four input trigger events to occur before responding with an output event.

The Model DMM7510 has two event blenders that you can program.

Event blender operations

You can use event blenders to perform logical AND or logical OR operations on trigger events. For example, trigger events can be triggered when either a manual trigger or external input trigger is detected.

When AND operation is selected, the event blender generates an event when an event is detected on all of the assigned stimulus inputs.

When OR operation is selected, the event blender generates an event when an event is detected on any one of the four stimulus inputs.

You set the event blender operation using remote commands.

Using SCPI commands:

Send the command `:TRIGger:BLENder<n>:MODE`.

Set the command to `OR` or `AND`.

Using TSP commands:

Send the command `trigger.blender[*N*].orenable`.

Setting the command to `true` enables OR operation; setting it to `false` enables AND operation.

Assigning blender trigger events

Each event blender has four stimulus inputs. You can assign a different trigger event to each stimulus input.

You set the blender stimulus events using remote commands. See the command descriptions for the list of events that you can assign.

Using SCPI commands:

Send the command `:TRIGger:BLENder<n>:STIMulus<m>`.

Using TSP commands:

Send the command `trigger.blender[*N*].stimulus[*M*]`.

Trigger blender action overruns

The event blenders can generate action overruns.

When the event blender operation is set to AND, overruns occur when a second event on any of its inputs is detected before an output event is generated.

When the operation is set to OR, overruns occur when two events are detected simultaneously.

Use the status model to monitor for the occurrence of action overruns. For details, see the Status model (on page C-1).

Interactive triggering

Interactive triggering is only available if you are using the TSP command set.

If you need more control of triggering than you can get using a trigger model, you can use interactive triggering to enable your system to generate and detect trigger events anywhere in the test flow. Interactive triggering is typically used in the context of TSP script operation. For example, interactive triggering can be used when you need to implement conditional branching to other test setups based on recent measurements.

All of the Model DMM7510 trigger objects have built-in event detectors that monitor for trigger events. The event detector only monitors events generated by that object. They cannot be configured to monitor events generated by any other trigger object.

You can use the wait() function of the trigger object to cause the instrument to suspend command execution until a trigger event occurs or until the specified timeout period elapses. For example, use trigger.blender[*N*].wait(*timeout*) to suspend command execution until an event blender generates an event, where *N* is the specific event blender and *timeout* is the timeout period. After executing the wait() function, the event detector of the trigger object is cleared.

The following programming example illustrates how to suspend command execution while waiting for various events to occur:

```
-- Wait up to 60 seconds for timer 1 to complete its delay.
trigger.timer[1].wait(60)
-- Wait up to 30 seconds for input trigger to digital I/O line 5.
trigger.digin[5].wait(30)
```

You can use some trigger objects to generate output triggers on demand. These trigger objects are the external I/O line, digital I/O lines, the TSP-Link synchronization lines, and the LAN.

The programming example below generates output triggers using the assert function of the trigger object.

```
-- Generate a 20 us pulse on digital I/O line 3.
digio.line[3].mode = digio.MODE_TRIGGER_OUT
trigger.digout[3].pulsewidth = 20e-6
trigger.digout[3].assert()
-- Generate a rising edge trigger on TSP-Link sync line 1.
tsplink.line[1].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN
trigger.tsplinkin[1].edge = trigger.EDGE_RISING
trigger.tsplinkout[1].logic = trigger.LOGIC_POSITIVE
trigger.tsplinkout[1].assert()
-- Generate a LAN trigger on LAN pseudo line 6.
-- Note that connection parameters and commands that
-- establish a connection are not shown.
trigger.lanout[6].assert()
```

Use the release function to allow the hardware line to output another external trigger when the pulse width is set to 0.

Setting the pulse width to 0 results in an indefinite length pulse when the assert function is used to output an external trigger. When an indefinite length pulse is used, the release function must be used to release the line before another external trigger can be output.

The release function can also be used to release latched input triggers when the hardware line mode is set to synchronous. In synchronous mode, the receipt of a falling edge trigger latches the line low. The release function releases this line high in preparation for another input trigger.

The programming example below illustrates how to output an indefinite external trigger.

```
-- Set digio line 1 to output an indefinite external trigger.
digio.line[1].mode = digio.MODE_TRIGGER_OUT
trigger.digout[1].logic = trigger.LOGIC_NEGATIVE
trigger.digout[1].pulsewidth = 0
trigger.digout[1].assert()
-- Release digio line 1.
trigger.digout[1].release()
-- Output another external trigger.
trigger.digout[1].assert()
```

For information about hardware lines, see Digital I/O lines (on page 3-47) , External I/O (on page 3-55), and Triggering using TSP-Link synchronization lines (on page 3-107).

The programming example below checks and responds to detector overruns.

```
testOver = trigger.digin[4].overrun
if testOver == true then
   print("Digital I/O overrun occurred.")
end
```

Trigger model

The trigger model controls the sequence in which measure actions occur. The Model DMM7510 trigger model is flexible, allowing you to control as much or as little as needed for your measurement application.

When you are setting up a trigger model, you can choose the following options:

- Wait for an event to occur before taking another measurement
- Notify other equipment and times that an event has occurred
- Wait for another piece of equipment to signal completion
- Use measure configuration lists to apply different measure settings dynamically during trigger model operation
- Specify delays between events and measurements
- Store measurements into a given buffer until an event occurs, then switch to another buffer
- Conditionally take actions based on whether the measurement falls within set limits

Additional options are detailed in the following sections.

The Model DMM7510 includes predefined trigger models to allow you to quickly implement a trigger model. You can also set up your own trigger models.

Trigger model building blocks

Each trigger model consists of building blocks that can be combined to create the trigger model. The building blocks can be combined from the front panel or by sending remote commands. You can connect a maximum of 63 building blocks as needed to control the instrument.

You can combine trigger model building blocks as you would construct a flow chart diagram. Trigger models are created using four fundamental building blocks:

- Wait: Waits for an event to occur before the flow continues
- Action: Starts an action in the instrument, such as making a measurement or clearing a buffer
- Notify: Notifies other equipment or timers that an event has occurred
- Branch: Branches when a condition has been satisfied

Each type of building block is described in the following topics.

Wait building block

The wait building block causes the trigger model to stop and wait for an event or set of events to occur before continuing. You can specify up to three events for each wait block.

The event can occur before the trigger model reaches the wait block. If the event occurs after the trigger model starts but before the trigger model reaches the wait block, the trigger model records the event. When the trigger model reaches the wait block, it executes the wait block without waiting for the event to happen again.

The instrument clears the memory of the recorded event when the trigger model exits the wait block.

You can have up to eight wait and branch on event blocks (in any combination) in a trigger model.

All items in the list are subject to the same action — you cannot combine AND and OR logic in a single wait block.

When you select the Wait block, the following options are available.

Setting	Description
Event 1	An event that must occur before the trigger block will continue.
Event Logic	Optional. Determines if all of the defined events must occur or if at least one of the events must occur. Select **AND** if all of the defined event must occur. Select **OR** if at least one of the event must occur.
Event 2	Optional. An event that must occur before the trigger block will continue.
Event 3	Optional. An event that must occur before the trigger block will continue.

The event can be any of the events described in the following table.

Event	Description
Digital Input	Line edge detected on a digital input line. When you select this option, you will also select the digital input to monitor. After selecting the digital input line, press **Config** to select the type of edge (falling, rising, or either).
TSP-Link Input	Line edge detected on a TSP-Link synchronization line. When you select this option, you will also select the TSP-Link line to monitor. After selecting the TSP-Link line, press **Config** to select the type of edge (falling, rising, or either).
Timer	A trigger timer expired. When you select this option, you will also select the timer to monitor. After selecting the timer, press **Config** to select the delay time, count, or start time.
Command	A command interface trigger: • Any remote interface: *TRG • GPIB only: GET bus command • VXI-11: VXI-11 command device_trigger
Display TRIGGER Key	Front-panel TRIGGER key press.
External In Trigger	Use a pulse from the external I/O. When you select this option, press **Config** to select the type of edge (falling, rising, or either).
Analog Trigger	Use the analog trigger.
LAN In Trigger	A LXI trigger packet is received on LAN trigger object. When you select this option, you will also select the LAN trigger to monitor. After you select the line, press **Config** to select the type of edge (falling, rising, or either).
Blender	Wait for the events set by an event blender.
None	No trigger event. The trigger model will not run if the wait event is set to none.

If you need to set up the trigger model to wait for an event under some conditions but not others, you can use a branch block. For information, see

Action building blocks

The action building blocks start an action in the instrument, such as making a measurement or clearing a buffer.

Measure building block

When trigger model execution reaches the block:

1. The instrument begins making a measurement.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurement into the specified reading buffer.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and measurements continue in the background until the trigger model execution reaches another measure block, a wait block, or until the trigger model ends.

There is a 2 μs delay after the block makes the last measurement in the count.

The measure block must be used with a measure function. The trigger model will not run if a digitize function is selected.

When you select the measure block, the following options are available.

Setting	Description
Reading Buffer	Optional. The name of the buffer where the measurement is stored.
Count	Optional. Specifies the number of readings to make before moving to the next block in the trigger model.

Digitize building block

When trigger model execution reaches the block:

1. The instrument begins digitizing measurements.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurements into the specified reading buffer.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all digitize measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and digitizing continues in the background until the trigger model execution reaches another digitize block or until the trigger model ends. The digitize block completes and moves on to the next block at the end of the aperture time, not the sample rate time.

For example, if there are two readings at a sample rate of 20,000 samples per second (50 µs apart) with an aperture of 1 µs, with a delay of 100 µs, the starts at 51 µs. The first reading occurs at 0 µs and the second starts at 50 µs, but it is completed at 51 µs because the aperture is only 1 µs. If the aperture is set to Auto, the first reading is at 0 µs, the second starts at 50 µs, and the delay starts at 100 µs.

The digitize block must be used with a digitize function. The trigger model will not run if a measure function is selected.

A trigger model that contains a digitize block may appear to hang in the wait block because it is making many measurements in one block.

There is a 2 µs delay after the digitize block makes the last measurement in the count.

When you select the digitize block, the following options are available.

Setting	Description
Reading Buffer	The name of the buffer where the measurement is stored
Count	Specifies the number of readings to make before moving to the next block in the trigger model

Constant delay building block

When trigger model execution reaches a delay building block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made, and if any previously executed block started infinite measurements, they also continue to be made.

This delay waits for the set amount of delay time to elapse before proceeding to the next block in the trigger model.

If other delays have been set, this delay is in addition to the other delays.

When you select the Constant Delay block, the following option is available.

Setting	Description
Delay	The amount of time to delay in seconds

Dynamic delay building block

When trigger model execution reaches a dynamic delay building block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made.

Each measure and digitize function can have up to 5 unique user delay times (M1 to M5). The delay time is set by the user-delay command. This command is only available over a remote interface. If you are using SCPI, the user delay command is [:SENSe[1]]:<function>:DELay:USER<n> (on page 6-77). If you are using TSP, it is dmm.measure.userdelay[N] (on page 8-194).

This delay can be different for every point in a configuration list. This makes it possible to have a delay that changes as a configuration list progresses.

When you select the Dynamic Delay block, the following option is available.

Setting	Description
Delay Type	The user delay to recall.

Buffer clear building block

When trigger model execution reaches the buffer clear trigger block, the instrument empties the specified reading buffer. The specified buffer can be the default buffer or a buffer that you defined.

If you are clearing a user-defined reading buffer, you must create the buffer before you define this block.

For more information about reading buffers, refer to Reading buffers (on page 3-12).

When you select the buffer clear block, the following option is available.

Setting	Description
Reading Buffer	The name of the buffer to clear

Config list next building block

The config list next building block recalls the settings at the next index point of a configuration list.

When trigger model execution reaches a configuration recall next building block, the settings at the next index point in the specified configuration list are restored.

The first time the trigger model encounters this block for a specific configuration list, the first index point is recalled. Each subsequent time this block is encountered, the settings at the next index point in the configuration list are recalled and take effect before the next step executes. When the last index point in the list is reached, it returns to the first point.

The configuration list must be defined before you can use this building block.

When you select the Config List block, the following option is available.

Setting	Description
Config List	The name of the configuration list from which to recall the next index point.

Config list prev building block

The config list prev building block defines a trigger model block that recalls the settings stored at the previous index point in a configuration list.

The configuration list previous index trigger block type recalls the previous index point in a configuration list. It configures the settings of the instrument based on the settings at that index. The trigger model executes the settings at that index before the next block is executed.

The first time the trigger model encounters this block, the last index in the configuration list is recalled. Each subsequent time trigger model execution reaches a configuration list previous block for this configuration list, it goes backward one index. When the first index in the list is reached, it goes to the last index in the configuration list.

You must create the configuration list before you can define it in this building block.

When you select the config list prev block, the following option is available.

Setting	Description
Config List	The name of the configuration list from which to recall the previous index point.

Config list recall building block

The configuration list recall building block loads a specific index from a configuration list.

All parameters that are defined in the configuration point are changed and take effect before this block completes and the next block begins.

In most cases, you should load the first configuration index at the beginning of the trigger model to ensure that the correct initial state is set and that the trigger model is repeatable.

You must define the configuration list before you can define it in this building block.

When you select the config list recall block, the following options are available.

Setting	Description
Config List	The name of the configuration block to recall the index point from
Recall Index	The index point to recall

Digital input/output building block

The digital I/O building block defines a trigger model block that sets the lines on the digital I/O port high or low.

To set the lines on the digital I/O port high or low, you can send an output line bit pattern. The pattern can be specified as an integer value, or, if you are using the TSP command set, a six-bit binary or hexadecimal. The least significant bit maps to digital I/O line 1 and the most significant bit maps to digital I/O line 6.

The output line bit mask defines the bits in the pattern that are driven high or low. If the bit for a line is set to 1, the line is driven high. If the bit is set to 0, the line is driven low. A binary 1 in the bit mask indicates that the corresponding I/O line should be driven according to the bit pattern. To drive all lines, specify all ones (63).

For this building block to work as expected, make sure you configure the trigger type and line state of the digital line for use with the trigger model (use the digital line mode command). The digital line settings are only available through remote commands.

When you select the digital I/O block, the following options are available.

Setting	Description
Out Line Pattern	Sets the value that specifies the output line bit pattern (0 to 63)
Out Line Mask	Sets the output line bit mask (0 to 63)

Log event building block

This block allows you to log an event in the event log when trigger model execution reaches this block. You can also force the trigger model to abort with this block. When the trigger model executes the block, the defined event is logged. If the abort option is selected, the trigger model is also aborted immediately.

You can define the type of event (information, warning, abort model, or error). All events generated by this block are logged in the event log. Warning and error events are also displayed in a popup on the front-panel display.

Note that using this block too often in a trigger model could overflow the event log. It may also take away from the time needed to process more critical trigger model blocks.

When you select the Log Event block, the following options are available.

Setting	Description
Event Type	The event number or type: • Abort Model: Stop the trigger model and log a warning message • Information N: Logs an information message in the event log • Warning N: Logs a warning in the event log • Error N: Logs an error in the event log Where N is 1 to 4; you can define up to four of each type
Message	A message that you define

Notify building block

When trigger model execution reaches a notify block, the instrument generates a trigger event and immediately continues to the next block.

Other commands can reference the event that the notify block generates. This assigns a stimulus somewhere else in the system. For example, you can use the notify event as the stimulus of a hardware trigger line, such as a digital I/O line.

Setting up the notify block using the front panel:

When you set up the notify blocks using the front panel, you select the line or timer to notify. You also set specifics regarding the line or timer. The stimulus and logic for input and output lines are set up automatically. The notify event number is also set automatically and is displayed at the bottom of the Notify definition screen.

When the trigger model executes a notify block, the instrument generates the SCPI event `NOTify<n>` or TSP event `trigger.EVENT_NOTIFYN`. You can assign this event to a command that takes an event. For example, if you want a notify block to trigger a digital I/O line, insert a notify block into the trigger model, assign it a notify event and then connect it to the stimulus of the digital I/O line that you want to drive with the notify event.

If you define a LAN trigger from the front panel, you are asked if you want to initiate the LAN connection. You must initiate the connection to use the LAN triggers.

Setting up the notify block using remote commands:

When you set up the notify block using remote commands, you define the notify event number. You need to set up the lines that use the notify event as a stimulus as separate commands.

In the following example, you define trigger model block 5 to be the notify 2 event. You can then assign the notify 2 event to be the stimulus for digital output line 3. To do this, send the following commands in SCPI:

```
:TRIG:BLOC:NOT 5, 2
:TRIG:DIG3:OUT:STIMulus NOTify2
```

In TSP, send the commands:

```
trigger.model.setblock(5, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFY2)
trigger.digout[3].stimulus = trigger.EVENT_NOTIFY2
```

If digital I/O line 3 is connected to another instrument, this causes the trigger execution to wait for the other instrument to indicate that it is ready.

Front panel options

When you select the Notify block from the front panel, the following options are available.

Setting	Description
Notify	The line or timer that is notified: • Digital Out Line: Select the digital output event (line 1 through line 6) • TSP-Link Out Line: Select the TSP-Link output event (line 1 through 3) • Timer: Select the timer event (timer 1 through 4) • External Out • LAN Out: Select the LAN output line (line 1 to 8)
Config	The available settings depend on the Notify selection: • For the digital, TSP-Link, external, and LAN out lines, select the pulse logic (negative or positive) • For the timer, select either delay or start time

Branching building blocks

A branch block goes to a trigger block other than the sequential execution block. For example, if you need to set up the trigger model to wait for an event under some conditions but not others, you can use a branch block to define when the wait block is enabled. You can use the Branch Once block to create a bypass and skip the wait block the first time the trigger model runs. This makes it possible to avoid deadlock when multiple instruments are being synchronized and each one is waiting for notification from the other one to start the trigger model.

Loop Counter building block

When trigger model execution reaches a loop counter block, it goes to a specified block until the count value is reached. When the counter exceeds the count value, trigger model execution ignores the branch, continues to the next building block in the sequence, and resets the counter.

The counter is reset to 0 when the trigger model starts. It is incremented each time trigger model execution reaches the counter block.

If you are using remote commands, you can query the counter. The counter is incremented immediately before the branch compares the actual counter value to the set counter value. Therefore, the counter is at 0 until the first comparison. When the trigger model reaches the set counter value, branching stops and the counter value is one greater than the setting.

When you select the Loop Counter block, the following options are available.

Setting	Description
Target Count	The number of times to repeat
Branch to Block	The block number to execute when the counter is less than the Target Count value

Constant Limit building block

The Branch Constant Limit block defines a trigger model block that branches to a block outside the normal trigger model flow if a measurement meets preset criteria.

The measurement block must be a measurement building block that occurs in the trigger model before the branch-on-constant-limits block. The last measurement from a measurement building block is used.

If the limit A is more than the limit B, the instrument automatically swaps the values so that the lesser value is used as the lower limit.

You can use this block to create a binning application by having the block branch to a digital I/O block, followed by a branch always block. Multiple tests can be chained together by repeating this.

NOTE

To use limits that vary programmatically, use the branch-on-dynamic-limits block.

When you select the Constant Limit block, the following options are available.

Setting	Description
Limit Type	How the limits are compared: • Inside: The measurement is inside the values set by limits A and B; limit A must be the low value and Limit B must be the high value • Above: The measurement is above the value set by limit B; limit A must be set, but is ignored when this type is selected • Below: The measurement is below the value set by limit A; limit B must be set, but is ignored when this type is selected • Outside: The measurement is outside the values set by limits A and B; limit A must be the low value and Limit B must be the high value
High Limit	The upper limit that the measurement is compared against. If the type is set to: • Inside: The high limit that the measurement is compared against • Above: The measurement must be above this value • Below: This value is ignored • Outside: The high limit that the measurement is compared against
Low Limit	The lower limit that the measurement is compared against. If the type is set to: • Inside: The low limit that the measurement is compared against • Above: This value is ignored • Below: The measurement must be below this value • Outside: The low limit that the measurement is compared against
Branch to Block	The block number to execute when the measurement meets the defined criteria
Measure Block	The block number of the measurement block that makes the measurement to be compared; from the front panel, you can set to Previous to use the previous measure or digitize block

Dynamic Limits building block

The branch-on-dynamic-limits building block defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

When you define this building block, you set:

- The type of limit (above, below, inside, or outside the limit values)
- The limit number (you can have 1 or 2 limits)
- The building block to go to if the measurement meets the criteria
- The building block that makes the measurement that is compared to the limits; the last measurement from that building block is used

There are two user-defined limits: limit 1 and limit 2. Both include their own high and low values, which are set using the front-panel Calculations limit settings or through commands. The results of these limit tests are recorded in the reading buffer that accompanies each stored reading.

Limit values are stored in the measure configuration list, so you can use a configuration list to step through different limit values.

The measure or digitize block must occur in the trigger model before the branch-on-dynamic-limits block. If no measure or digitize block is defined, the measurement from the previous measure or digitize block is used. If no previous measure or digitize block exists, an error is reported.

When you select the Dynamic Limit block, the following options are available.

Setting	Description
Limit Type	How the limits are compared: • Inside: The measurement is within the limits • Above: The measurement is above the high limit • Below: The measurement is below the low limit • Outside: The measurement is outside the limits
Limit Number	The limit that is used for this block, 1 or 2
Branch to Block	The block number to execute when the measurement meets the defined criteria
Measure Block	The block number of the measurement block that makes the measurement to be compared; from the front panel, you can set to Previous to use the previous measure or digitize block

Once building block

When the trigger model reaches a branch-once building block, it goes to a specified building block the first time it is encountered in the trigger model. If it is encountered again, the trigger model ignores the block and continues in the normal sequence.

You can use this block to create a bypass. For example, you might place a branch-once block before a wait block to skip the wait block on the first pass of the trigger model.

The once block is reset when the trigger model reaches the idle state. Therefore, the branch-once block will always execute the first time the trigger model encounters this block.

When you select the Once block, the following option is available.

Setting	Description
Branch to Block	The block number to execute the first time the trigger model reaches the Once block

Once excluded building block

The branch-once-excluded building block is ignored the first time the trigger model encounters it. After the first encounter, the trigger model goes to the specified branching block.

The branch-once-excluded block is reset when the trigger model starts or is placed in idle.

When you select the once excluded block, the following option is available.

Setting	Description
Branch to Block	The block number to execute after the first time the trigger model reaches the once excluded block.

Delta building block

The branch on delta building block defines a trigger model block that goes to a specified block if the difference of two measurements meets preset criteria.

This block calculates the difference between the last two measurements from a measure or digitize block. It subtracts the most recent measurement from the previous measurement.

The difference between the measurements is compared to the target difference. If the difference is less than the target difference, the trigger model goes to the specified branching block. If the difference is more than the target difference, the trigger model proceeds to the next block in the trigger block sequence.

If you do not define the measure or digitize block, it will compare measurements of a measure or digitize block that precedes the branch delta block. For example, if you have a measure block, a wait block, another measure block, another wait block, and then the branch delta block, the delta block compares the measurements from the second measure block. If a preceding measure or digitize block does not exist, an error occurs.

When you select the Delta block, the following options are available.

Setting	Description
Target Delta	The value against which the block compares the difference between the measurements
Branch to Block	The block number of the trigger model block to execute when the difference between the measurements is less than or equal to the Target Delta
Measure Block	The block number of the measure or digitize block that makes the measurements to be compared; if this is 0 or undefined, the trigger model uses the previous measure or digitize block

On event building block

The branch-on-event building block branches to a specified block when a specified trigger event occurs. If the trigger event has not yet occurred when trigger model execution reaches the branch-on-event block, the trigger model continues to execute the blocks in the normal sequence. After the trigger event occurs, the next time trigger model execution reaches the branch-on-event block, it goes to the branching block.

Trigger events are reset when the trigger model is at the start block, so only events that occur after you start trigger model execution are detected by the branch-on-event block. The event is also reset after trigger model execution completes the branching block.

You can have up to eight branch-on-event blocks in a trigger model.

When you select the branch-on-event block, the following options are available.

Setting	Description
Event	The event that causes this block to branch
Branch to Block	The block number to execute when the specified event occurs

The event can be any of the events described in the following table.

Event	Description
Digital Input	Line edge detected on a digital input line. When you select this option, you will also select the digital input to monitor. After selecting the digital input line, press **Config** to select the type of edge (falling, rising, or either).
TSP-Link Input	Line edge detected on a TSP-Link synchronization line. When you select this option, you will also select the TSP-Link line to monitor. After selecting the TSP-Link line, press **Config** to select the type of edge (falling, rising, or either).
Timer	A trigger timer expired. When you select this option, you will also select the timer to monitor. After selecting the timer, press **Config** to select the delay time, count, or start time.
Command	A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`
Display TRIGGER Key	Front-panel TRIGGER key press.
External In Trigger	Use a pulse from the external I/O. When you select this option, press **Config** to select the type of edge (falling, rising, or either).
Analog Trigger	Use the analog trigger.
LAN In Trigger	A LXI trigger packet is received on LAN trigger object. When you select this option, you will also select the LAN trigger to monitor. After you select the line, press **Config** to select the type of edge (falling, rising, or either).
Blender	Wait for the events set by an event blender.
None	No trigger event. The trigger model will not run if the wait event is set to none.

For information on trigger events, see <u>Using trigger events to start actions in the trigger model</u> (on page 3-97).
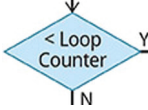
Always building block

When the trigger model reaches a branch-always building block, it goes to the building block that you specified.
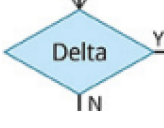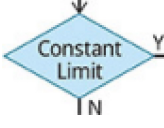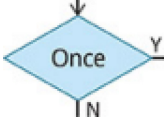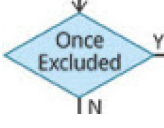
When you select the always block, the following option is available.
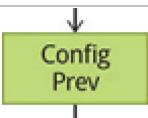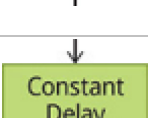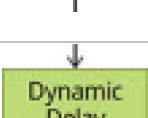
Setting	Description
Branch to Block	The block number to execute when the trigger model reaches the always block

Trigger block summary
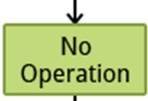
Front-panel icon	SCPI command TSP command	Block description
Not applicable	:TRIGger:BLOCk:LIST? (on page 6-187) trigger.model.getblocklist() (on page 8-280)	This returns the settings for all trigger model building blocks
Always	:TRIGger:BLOCk:BRANch:ALWays (on page 6-170) trigger.model.setblock() — trigger.BLOCK_BRANCH_ALWAYS (on page 8-296)	This defines a trigger model block that always goes to a specific block
< Loop Counter	:TRIGger:BLOCk:BRANch:COUNter (on page 6-171) trigger.model.setblock() — trigger.BLOCK_BRANCH_COUNTER (on page 8-297)	This defines a trigger model block that branches to a specified block a specified number of times
Not applicable	:TRIGger:BLOCk:BRANch:COUNter:COUNt? (on page 6-172) trigger.model.getbranchcount() (on page 8-281)	This returns the count value of the trigger model counter block
Reset Branch Count	:TRIGger:BLOCk:BRANch:COUNter:RESet (on page 6-172) trigger.model.setblock() — trigger.BLOCK_RESET_BRANCH_COUNT (on page 8-319)	This creates a block in the trigger model that resets a branch counter to 0

Trigger block summary

Delta	:TRIGger:BLOCk:BRANch:DELTa (on page 6-173) trigger.model.setblock() — trigger.BLOCK_BRANCH_DELTA (on page 8-298)	This defines a trigger model block that goes to a specified block if the difference of two measurements meets preset criteria
On Event	:TRIGger:BLOCk:BRANch:EVENt (on page 6-174) trigger.model.setblock() — trigger.BLOCK_BRANCH_ON_EVENT (on page 8-302)	This branches to a specified block when a specified trigger event occurs
Constant Limit	:TRIGger:BLOCk:BRANch:LIMit:CONStant (on page 6-176) trigger.model.setblock() — trigger.BLOCK_BRANCH_LIMIT_CONSTANT (on page 8-299)	This defines a trigger model block that branches to a block outside the normal trigger model flow if a measurement meets preset criteria
Dynamic Limit	:TRIGger:BLOCk:BRANch:LIMit:DYNamic (on page 6-177) trigger.model.setblock() — trigger.BLOCK_BRANCH_LIMIT_DYNAMIC (on page 8-301)	This defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria
Once	:TRIGger:BLOCk:BRANch:ONCE (on page 6-178) trigger.model.setblock() — trigger.BLOCK_BRANCH_ONCE (on page 8-304)	This causes the trigger model to branch to a specified building block the first time it is encountered in the trigger model
Once Excluded	:TRIGger:BLOCk:BRANch:ONCE:EXCLuded (on page 6-179) trigger.model.setblock() — trigger.BLOCK_BRANCH_ONCE_EXCLUDED (on page 8-305)	This causes the trigger model to go to a specified building block every time the trigger model encounters it, except for the first time

Front-panel icon	SCPI command TSP command	Block description
Buffer Clear	:TRIGger:BLOCk:BUFFer:CLEar (on page 6-180) trigger.model.setblock() — trigger.BLOCK_BUFFER_CLEAR (on page 8-306)	This defines a trigger model block that clears the reading buffer
Config Next	:TRIGger:BLOCk:CONFig:NEXT (on page 6-180) trigger.model.setblock() — trigger.BLOCK_CONFIG_NEXT (on page 8-307)	This recalls the settings at the next index point of a configuration list
Config Prev	:TRIGger:BLOCk:CONFig:PREVious (on page 6-181) trigger.model.setblock() — trigger.BLOCK_CONFIG_PREV (on page 8-308)	This defines a trigger model block that recalls the settings stored at the previous index point in a configuration list
Config Recall	:TRIGger:BLOCk:CONFig:RECall (on page 6-182) trigger.model.setblock() — trigger.BLOCK_CONFIG_RECALL (on page 8-309)	This recalls the system settings that are stored in a configuration list
Constant Delay	:TRIGger:BLOCk:DELay:CONStant (on page 6-183) trigger.model.setblock() — trigger.BLOCK_DELAY_CONSTANT (on page 8-310)	This adds a constant delay to the trigger model
Dynamic Delay	:TRIGger:BLOCk:DELay:DYNamic (on page 6-184) trigger.model.setblock() — trigger.BLOCK_DELAY_DYNAMIC (on page 8-311)	This adds a delay to the execution of the trigger model

Trigger block summary

Front-panel icon	SCPI command TSP command	Block description
Set Digital I/O	:TRIGger:BLOCk:DIGital:IO (on page 6-185) trigger.model.setblock() — trigger.BLOCK_DIGITAL_IO (on page 8-312)	This trigger model block that sets the lines on the digital I/O port high or low
Digitize	:TRIGger:BLOCk:DIGitize (on page 6-186) trigger.model.setblock() — trigger.BLOCK_DIGITIZE (on page 8-313)	This defines a trigger block that makes a measurement using a digitize function
Log Event	:TRIGger:BLOCk:LOG:EVENt (on page 6-188) trigger.model.setblock() — trigger.BLOCK_LOG_EVENT (on page 8-315)	This allows you to log an event in the event log when the trigger model is running
Measure	:TRIGger:BLOCk:MEASure (on page 6-189) trigger.model.setblock() — trigger.BLOCK_MEASURE (on page 8-316)	This defines a trigger block that makes a measurement using a measure function
No Operation	:TRIGger:BLOCk:NOP (on page 6-190) trigger.model.setblock() — trigger.BLOCK_NOP (on page 8-317)	This creates a placeholder that performs no action in the trigger model
Notify	:TRIGger:BLOCk:NOTify (on page 6-191) trigger.model.setblock() — trigger.BLOCK_NOTIFY (on page 8-318)	This defines a trigger model block that generates a trigger event and immediately continues to the next block
Wait	:TRIGger:BLOCk:WAIT (on page 6-192) trigger.model.setblock() — trigger.BLOCK_WAIT (on page 8-320)	This defines a trigger model block that waits for an event before allowing the trigger model to continue

Predefined trigger models

The Model DMM7510 includes predefined trigger models for common applications. You can use these predefined trigger models without changing them, or you can modify them to meet the needs of your application.

The predefined trigger models include:

- **Empty:** Clears the present trigger model.
- **ConfigList:** Creates a trigger model that loads a configuration list. At each configuration list point, a measurement is made. The list is iterated until every point in the configuration list has been loaded.
- **ExtDigTrigger:** Creates a trigger model that waits on an input line, delays, makes a measurement, and sends out a trigger on the output line a specified number of times.
- **SimpleLoop:** Creates a trigger model that makes a specified number of readings. A count parameter defines the number of readings.
- **DurationLoop:** Creates a trigger model that makes continuous measurements for a specified amount of time.
- **LoopUntilEvent:** Creates a trigger model that makes continuous measurements until a specified event occurs.
- **GradeBinning:** Creates a trigger model that successively measures components and compares their readings to high or low limits to grade components.
- **SortBinning:** Creates a trigger model that successively measures components and compares their readings to high or low limits to sort components.
- **Keithley2001:** Creates a multi-layer trigger model with parameters for bypass, count, and delay.

Using a predefined trigger model

Before starting the trigger model, you need to set up your instrument for testing, including the measure or digitize settings and configuration lists.

When you load a predefined trigger model, the instrument overwrites any existing trigger models.

Using the front panel:

1. Press the **MENU** key.
2. Under Trigger, select **Templates**.
3. Next to Templates, select the trigger model template to use.
4. If the template you select has additional settings, you can use the default values or make any necessary changes to the settings.
5. Press the **TRIGGER** key to initiate the trigger model. The trigger mode indicator shows the status of the trigger mode. See Trigger mode indicator (on page 2-15) for descriptions of the indicators.

Using SCPI commands:

See the descriptions of the `TRIGger:LOAD` commands for details on the options available for each predefined trigger model:

- :TRIGger:LOAD "ConfigList" (on page 6-210)
- :TRIGger:LOAD "DurationLoop" (on page 6-211)
- :TRIGger:LOAD "Empty" (on page 6-212)
- :TRIGger:LOAD "ExtDigTrigger" (on page 6-213)
- :TRIGger:LOAD "GradeBinning" (on page 6-214)
- :TRIGger:LOAD "LoopUntilEvent" (on page 6-218)
- :TRIGger:LOAD "Keithley2001" (on page 6-216)
- :TRIGger:LOAD "SimpleLoop" (on page 6-220)
- :TRIGger:LOAD "SortBinning" (on page 6-221)

Using TSP commands:

See the descriptions of the `trigger.model.load()` command for details on the options available for each predefined trigger model:

- trigger.model.load() — ConfigList (on page 8-282)
- trigger.model.load() — DurationLoop (on page 8-284)
- trigger.model.load() — Empty (on page 8-285)
- trigger.model.load() — ExtDigTrigger (on page 8-286)
- trigger.model.load() — GradeBinning (on page 8-287)
- trigger.model.load() — Keithley2001 (on page 8-289)
- trigger.model.load() — LoopUntilEvent (on page 8-290)
- trigger.model.load() — SimpleLoop (on page 8-292)
- trigger.model.load() — SortBinning (on page 8-294)

Using a predefined trigger model to develop a trigger model

The Model DMM7510 includes predefined trigger models that you can use as a starting point for developing your trigger model.

After modifying a predefined trigger model, you can save it in a saved setup for future use. See Saving setups (on page 2-148) for information on how to save a configuration.

Using the front panel:
1. Press the **MENU** key.
2. Under Trigger, select **Templates**. The TRIGGER MODEL TEMPLATES screen is displayed.
3. Next to Templates, select the trigger model to use.
4. If the template you select has additional settings, you can use the default values or make any necessary changes to the settings.
5. Select **EXIT** to return to the MENU screen.
6. Under Trigger, select **Configure**. The blocks for the predefined trigger model are displayed.
7. Choose or modify the blocks as needed. See Assembling trigger model building blocks (on page 3-92).
8. When the blocks are set up, select **EXIT** to return to the MENU screen.
9. Press the **TRIGGER** key to initiate the trigger model. The trigger mode indicator shows the status of the trigger mode. See Trigger mode indicator (on page 2-15) for descriptions of the indicators.

Assembling trigger model building blocks

This section describes the basic concepts you need to understand to assemble trigger model building blocks.

Sequencing trigger model building blocks

You can set up the trigger model building block from the front panel or by using remote commands.

Building blocks must be sequenced in order — you cannot skip numbers. When the trigger model completes the last block in the trigger model, the trigger model returns to idle. Idle is considered to be execution block 0. Branching to block 0 effectively stops the trigger model.

As the trigger model reaches each block, the action defined by that block is started and completed before the trigger model moves to the next block. Blocks do not overlap.

The trigger model steps through the building blocks in sequential order. You can set up branching blocks to allow nonsequential actions to occur. See Branching building blocks (on page 3-81) for detail on how to use the branching blocks.

If you skip block numbers, when you initiate the trigger model, the trigger model generates an event message that reports the missing block. You can view and delete the missing blocks on the front-panel TriggerFlow. If you delete them using the front-panel options, the remaining blocks are resequenced.

You can have up to 63 blocks in a trigger model.

Working with the trigger model

You can change existing trigger model blocks through the front panel or by sending a remote command. The block is redefined with the new parameters.

When you define the trigger model using remote commands, you can send blocks in any order. For example, you can define block 5 before defining blocks 1 to 4. However, you cannot run a trigger model with undefined blocks.

If you skipped a block, you can use the no operation block to define a block that will not affect the trigger model and save the effort of resequencing the other blocks. The no operation block is available through the remote commands only (SCPI command :TRIGger:BLOCk:NOP (on page 6-190) or TSP command trigger.model.setblock() — trigger.BLOCK_NOP (on page 8-317)).

Determining the structure of the existing trigger model

You can retrieve the existing trigger model structure from the front panel or by using remote commands.

Using the front panel:

1.   Press the **MENU** key.
2.   Under Trigger, select **Configure**. The trigger model is displayed.
3.   If trigger model is longer than one page, swipe the TriggerFlow block to scroll up or down.
4.   To view the settings for a block, select the block. The settings are displayed on the right.
5.   For a description of a setting, highlight the button and press **HELP**.

For additional information on the blocks, refer to the block descriptions under Trigger model building blocks (on page 3-73).

Using SCPI commands:

To retrieve the settings for all trigger model building blocks, send the command:

```
:TRIGger:BLOCk:LIST?
```

Using TSP commands:

To check the settings for a building block, send the command:

```
print(trigger.model.getblocklist())
```

NOTE

To retrieve the TSP code for trigger model blocks that were entered through the front panel, change the Event Log "Command" setting to On. Refer to Using the event log (on page 2-152) for additional information.

Improving the performance of a trigger model

To improve the performance of a trigger model:

- Reduce the number of blocks to less than 15.
- Do not use multiple reading buffers.
- Use four or fewer delay blocks.
- Use four or fewer measure or digitize blocks.
- Do not have multiple blocks waiting on the same event.
- Verify that constant delay blocks are set to less than 254 ms.
- Limit use of configuration list blocks.

Action overruns

An action overrun occurs when a trigger object receives a trigger event and is not ready to act on it. The action overruns of all trigger objects are reported in a command for the associated trigger object. See the appropriate sections on each trigger object for further details on conditions under which an object generates an action overrun.

Running the trigger model

You can run the trigger model when the instrument is controlled either locally or remotely. Note that if you change from remote to local control, the trigger model measurement method remains selected until you change it.

When you run the trigger model, the existing instrument settings are used for any actions unless you assigned configuration lists to the trigger model.

Trigger model operation is an overlapped process. This means that you can run other commands while a trigger model is running if they do not conflict with trigger model operation. For example, you can print the buffer contents, but you cannot change the measure function.

The initiate command is the overlapped command that starts the process. The command interface is available immediately after the instrument executes the initiate command so that other commands can be executed while the Trigger Model is running.

To change the measurement method, see Switching between measurement methods (on page 3-59).

Starting the trigger model

Using the front panel:

1. Press the front-panel **TRIGGER** key for 2 s. A dialog box displays the available trigger methods. The presently selected method is highlighted.
2. Select **Initiate Trigger Model**.
3. If the instrument is controlled remotely, a confirmation screen is displayed. Select **Yes** to change to front-panel control and start the trigger model.

Using SCPI commands:

Send the command:

```
:INITiate
```

Using TSP commands:

Send the command:

```
trigger.model.initiate()
```

Aborting the trigger model

You can stop the trigger model while it is in progress. When you stop the trigger model, all trigger model commands on the instrument are terminated.

Using the front panel:

Press the **TRIGGER** key for 2 s and select **Abort Trigger Model**.

Using SCPI commands:

Send the command:

```
:ABORt
```

Using TSP commands:

Send the command:

```
trigger.model.abort()
```

Checking the state of the trigger model

The trigger model can be in one of several states. The state is shown in the indicator bar on the Home screen of the instrument. You can also check the status using remote commands.

The following table describes the trigger model states. This table also describes the indicator that is shown on the front panel and the feedback you get from the remote interface.

Front panel indicator	Remote command feedback — SCPI	Remote command feedback — TSP	Description
CONT	Not available through remote interface	Not available through remote interface	Instrument is not using the trigger model; it is making measurements continuously
MAN	Not available through remote interface	Not available through remote interface	Instrument is not using trigger model; makes measurements when you press the front-panel TRIGGER key
IDLE	`IDLE` or `EMPTY`	`trigger.STATE_IDLE` or `trigger.STATE_EMPTY`	Trigger model is stopped or no block are defined
RUN	`RUNNING`	`trigger.STATE_RUNNING`	Trigger model is running
WAIT	`WAITING`	`trigger.STATE_WAITING`	The trigger model has been in the wait block for more than 100 ms
ABORT	`ABORTED`	`trigger.STATE_ABORTED`	The trigger model was stopped before it completed

Using the front panel

The state of the trigger model is indicated on the status bar with the indicators shown in the previous table.

Using SCPI commands:

Send the command:

```
:TRIGger:STATe?
```

The return shows the state and the block that was last executed.

Using TSP commands:

Send the command:

```
print(trigger.model.state())
```

The return shows the state and the block that was last executed.

Using trigger events to start actions in the trigger model

You can set up trigger blocks to respond to trigger events. Trigger events are signals that can be generated by the instrument or by other system components.

Sources of the trigger event signals can be:

- Front-panel TRIGGER key
- Notify trigger blocks
- Branch-on-event trigger blocks
- Command interface triggers
- Digital I/O lines
- TSP-Link synchronization lines
- LAN triggers
- Analog triggers
- External I/O triggers
- Event blenders, which combine other trigger events
- Trigger timers

For information about the options that are not specific to the trigger model, see

Trigger events

To use trigger events, you need to specify the event constant. The tables below show the constants for the trigger events in the system. You can use these events with instrument features such as trigger timers, trigger blocks, digital I/O lines, and external I/O lines.

Trigger events - SCPI command set

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: - Any remote interface: `*TRG` - GPIB only: GET bus command - VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Trigger events - TSP command set

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender N (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer N (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Using the TRIGGER key to generate an event

You can use the front-panel TRIGGER key to generate a trigger event.

For example, if you set a wait block to advance when the TRIGGER key is pressed, the trigger model will reach the wait block. If the TRIGGER key has already been pressed, the trigger model execution will continue. If the TRIGGER key has not been pressed, the trigger model execution is halted until the TRIGGER key is pressed.

To set a trigger block to respond to the front-panel key press:

• From the front panel: Set the event to be Display TRIGGER Key

• Using SCPI: Set the event to `DISPlay`

• Using TSP: Set the event to `trigger.EVENT_DISPLAY`

There are no action overruns for front-panel TRIGGER key events.

Respond to an event with a wait block

The wait building block causes the trigger model to stop and wait for an event or set of events to occur before continuing. You can specify up to three events for each wait block. The wait block can use any of the system trigger events. See .

To continue the trigger model, it must receive the trigger event that is defined for the wait block.

Using the branch-on-event trigger blocks

The branch-on-event building block goes to a branching block after a specified trigger event occurs. If the trigger event has not yet occurred when the trigger model reaches the branch-on-event block, the trigger model continues to execute the blocks in the normal sequence. After the trigger event occurs, the next time the trigger model reaches the branch-on-event block, it goes to the branching block.

If you set the branch event to none, an error is generated when you run the trigger model.

The branch-on-event block can use any system trigger events. See Trigger events (on page 3-97).

Limit testing and binning

The Model DMM7510 can be set up for limit testing and binning. It can perform simple benchtop limit testing using the front panel or sophisticated limit and binning operations using the trigger model and digital I/O to control external component-handling devices.

Some typical forms of limit testing include:

- Simple pass-or-fail testing
- Resistor grading: Inspect multiple limits until the first failure is received
- Resistor sorting: Inspect multiple limits until the first pass is received

For binning applications, you use limit testing to determine placement of tested parts. To set up the instrument to place the part in the correct bin, you do the following steps:

- Determine and record a bin number for later use
- Output a digital bit pattern to physically place the tested device in a bin
- If multiple tests are performed on the same part, determine when the part should be binned:
  - Bin the part as soon as it fails a test
  - Bin the part after all parameters are measured; bin according to the first failure or a combination of failures

Limit testing allows you to set high and low limit values. When the reading falls outside these limits, the instrument displays L1FAIL or L2FAIL.

The limit values are stored in volatile memory.

Limits are tested after any selected filter, relative offset, and math functions have been applied to the measurement.

The Model DMM7510 provides two binning trigger model templates to assist with setup, one for grading and one for sorting. Refer to Predefined trigger models (on page 3-90)

Limit testing using the front-panel interface

You can do pass or fail limit testing through the front panel. When limit testing and a test fails, the limit (#1 or #2) that failed is shown on the Home screen.

Using the front panel:
1. Press the **MENU** key.
2. Under Measure, select **Calculations**.
3. Set Limit 1 or Limit 2 to **On**.
4. Select **Config**.
5. The Auto Clear setting automatically clears the limit fail indicator when a new passing measurement is made. To turn this feature off, select **Off**.
6. Set the **Low Value**. If the measurement is below the Low Value, the limit failure indicator is displayed.
7. Set the **High Value**. If the measurement is above the High Value, the limit failure indicator is displayed.
8. The Audible setting determines if a beeper sounds when a measurement passes or fails. Set as needed.
9. Select **HOME** to return to the operating display.
10. Make a measurement. L1PASS is displayed if the measurement is in the limits; L1FAIL is displayed if the measurement is not in the limits.

An example of using limit testing to check resistors is described in the following topic.

Front-panel limit test

This example is set up to test a box of 100 Ω ±1 % and 100 Ω ±10 % resistors that you want to separate manually. You can change values as needed to adapt the test to your needs.

To set up the test:
1. Press the **FUNCTION** key.
2. Select **4W Res**.
3. Press the **MENU** key.
4. Under Measure, select **Calculations**.
5. Set Limit 1 and Limit 2 to **On**.
6. Select **Config** for Limit 1.
7. Set the Low Value to **90** Ω.
8. Set the High Value to **110** Ω.
9. Select **OK**.
10. Select **Config** for Limit 2.
11. Set the Low Value to **99** Ω.
12. Set the High Value to **101** Ω.
13. Select **OK**.

Run the test:

1. Press the **HOME** key.

2. Use 4-wire connections to connect the first resistor to the instrument.

3. Verify that the instrument is set to Continuous Measurement. If necessary, hold the **TRIGGER** key for 2 s and select **Continuous Measurement**.

4. Observe the measurements. If the resistor is inside the limits set for Limit 1, L1PASS is displayed. If the resistor is not within the limits, L1FAIL is displayed. If the resistor is in the limits set for Limit 2, L2PASS is displayed. If the resistor is not within the limits, L2FAIL is displayed. An example of a test that passed the L1 test but failed the L2 test is shown below.

Figure 127: Limit test pass and fail indicators

TSP-Link System Expansion Interface

Keithley Instruments TSP-Link[®] is a high-speed trigger synchronization and communication bus that test system builders can use to connect multiple instruments in a master and subordinate configuration. Once connected, all the instruments that are equipped with TSP-Link in a system can be programmed and operated under the control of the master instrument or instruments. This allows the instruments to run tests more quickly because they can be decoupled from frequent computer interaction. The test system can have multiple master and subordinate groups, which can be used to handle multi-device testing in parallel. Combining TSP-Link with a flexible programmable trigger model ensures speed.

Using TSP-Link, multiple instruments are connected and can be used as if they are part of the same physical unit for simultaneous multi-channel testing. The test system can be expanded to include up to 32 TSP-Link-enabled instruments.

TSP-Link functionality is only available when using the instrument front panel or the TSP commands to control the instrument. It is not available if you are using SCPI commands.

TSP-Link connections

The Model DMM7510 has three synchronization lines that are built into the TSP-Link connection. If you are using a TSP-Link network, you do not have to modify any connections.

Example connections for a TSP-Link system are shown in the following figure.

The TSP-Link connectors are on the rear panel of the instruments. All of the instruments in the system are connected in a sequence (daisy-chained) using LAN crossover cables.

Figure 128: TSP-Link connections
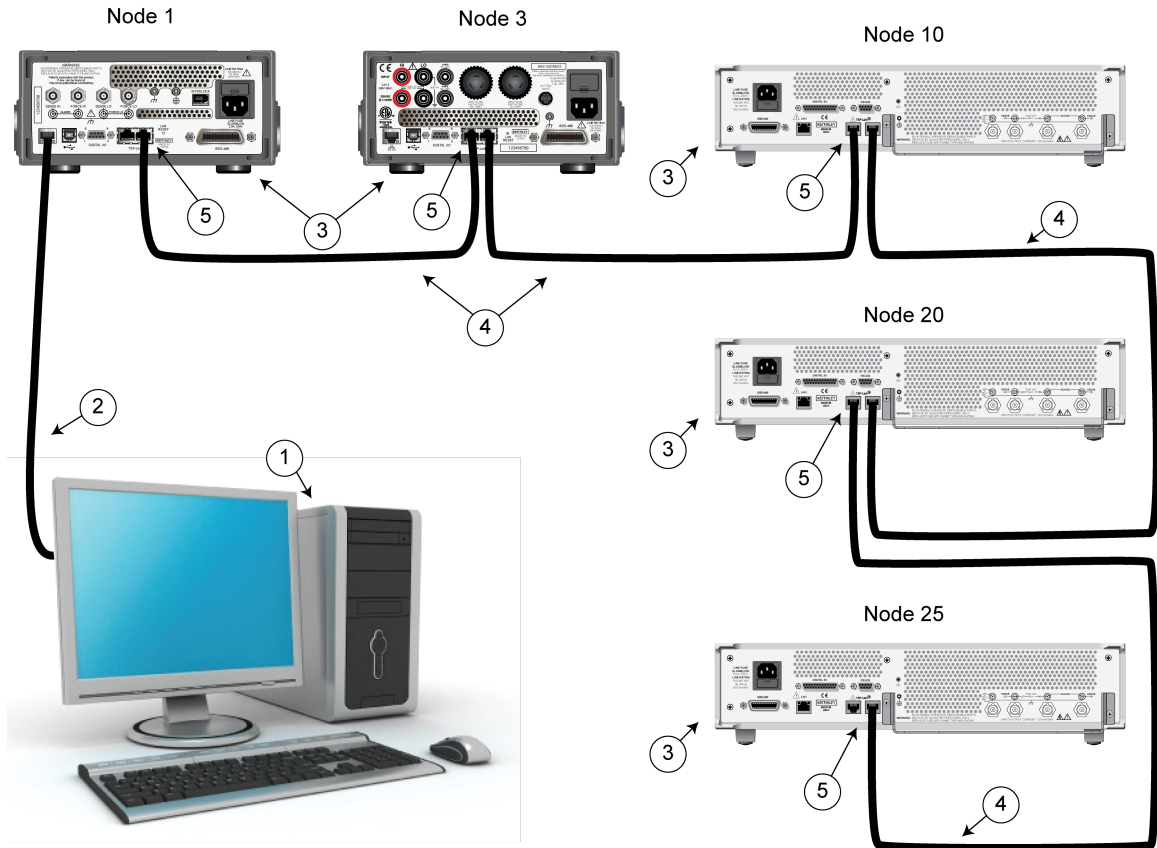
Item	Description	Notes
1	Controller	Optional. A computer is not needed for stand-alone systems.
2	Communication connection	Optional. Connection from controller to the master node through GPIB, LAN, or USB. Details about these computer communication connections are described in Remote communications interfaces (on page 2-61).
3	Nodes	You can have up to 32 nodes on the TSP-Link system. Each node must have a unique node number from 1 to 64.
4	LAN crossover cable	Type 5e category or higher; 3 meters (9.8 feet) maximum between nodes. Available from Keithley Instruments (Model CA-180-3A).
5	TSP-Link connections	Each instrument has two TSP-Link connectors. You can make the connection to either TSP-Link connection.

TSP-Link nodes

Each instrument or enclosure attached to the TSP-Link expansion interface is called a node. Each node must be identified with a unique node number. This identification is called a TSP-Link node number.

An individual node is accessed as `node[N]`, where `N` is the node number assigned to the node. You can access all TSP-accessible remote commands as elements of the specific node. The following attributes are examples of items you can access:

*   `node[N].model`: The product model number of the node.
*   `node[N].version`: The product version of the node.
*   `node[N].serialno`: The product serial number of the node.

Assigning node numbers

Each Model DMM7510 instrument is initially assigned as node 2. You can assign node numbers from 1 to 64. However, the system can only include 32 physical nodes.

The node number for each instrument is stored in its nonvolatile memory and remains in storage when the instrument is turned off.

You can assign a node number to an instrument using the front panel or by using a remote command.

To assign a node number using the front panel:

1.   Press the **MENU** key.
2.   Under System, select **Communication**. The SYSTEM COMMUNICATION window opens.
3.   Select the **TSP-Link** tab.
4.   Next to Node, set the TSP-Link address for this instrument.

To assign a node number using a remote command:

Set the `tsplink.node` attribute of the instrument:

```
tsplink.node = N
```

Where: `N` = 1 to 64

To determine the node number of an instrument, you can read the `tsplink.node` attribute by sending the following command:

```
print(tsplink.node)
```

The above `print` command outputs the node number. For example, if the node number is 1, a `1` is displayed.

Master and subordinates

In a TSP-Link® system, one of the nodes (instruments) is the master node and the other nodes are the subordinate nodes. The master node in a TSP-Link® system can control the other nodes (subordinates) in the system.

When any node transitions from local operation to remote operation, it becomes the master of the system. All other nodes also transition to remote operation and become its subordinates. When any node transitions from remote operation to local, all other nodes also transition to local operation, and the master/subordinate relationship between nodes is dissolved.

A TSP-Link system can be stand-alone or computer-based.

In a stand-alone system, scripts are loaded into the instruments. You can run a script from the front panel of any instrument (node) connected to the system. When a script is run, all nodes in the system go into remote operation. When the script is finished running, all the nodes in the system return to local operation, and the master/subordinate relationship between nodes is dissolved.

In a computer-based system, you can use a computer and a remote interface to communicate with a single node in the system. This node becomes the interface to the entire system. When a command is sent through this node, all nodes go into remote operation. The node that receives the command becomes the master and can control all of the other nodes, which become its subordinates. In a computer-based system, the master/subordinate relationship between nodes can only be dissolved by performing an abort operation. For more information about remote interfaces, see Remote communications interfaces (on page 2-61).

Initializing the TSP-Link system

The TSP-Link® system must be initialized after configuration changes. You need to initialize the system after you:

* Turn off power or reboot any instrument in the system
* Change node numbers on any instrument in the system
* Rearrange or disconnect the TSP-Link cable connections between instruments

If initialization is not successful, you can check the event log for error messages that indicate the problem. Some typical problems include:

* Two or more instruments in the system have the same node number
* There are no other instruments connected to the instrument performing the initialization
* One or more of the instruments in the system is turned off
* The actual number of nodes is less than the expected number

From the front panel:

1. Power on all instruments connected to the TSP-Link network.
2. Press the **MENU** key.
3. Under System, select **Communication**. The SYSTEM COMMUNICATION window opens.
4. Select the **TSP-Link** tab.
5. Select **Initialize**.

Using TSP commands:

To initialize the TSP-Link system, send the command:

```
tsplink.initialize()
```

To check the state of the TSP-Link system, send the command:

```
print(tsplink.state)
```

If initialization was successful, `online` is returned. If initialization was not successful, `offline` is returned.

Sending commands to TSP-Link nodes

You can send remote commands to any instrument on the TSP-Link system by adding `node[N].` to the beginning of the remote command, where N is the node number.

For example, to sound the beeper on node 10, you would send the command:

```
node[10].beeper.beep(2, 2400)
```

To send a command to the master, you can interact with it as if it were a single instrument.

Using the reset() command

Most TSP-Link® system operations target a single node in the system, but the `reset()` command affects the system as a whole by resetting all nodes to their default settings:

```
-- Reset all nodes in a TSP-Link system to their default state.
reset()
```

NOTE

Using the `reset()` command in a TSP-Link network differs from using the `tsplink.reset()` or `tsplink.initialize()` command. The `tsplink.reset()` or `tsplink.initialize()` command reinitializes the TSP-Link network and will turn off the output of any TSP-linked instrument; it may change the state of individual nodes in the system.

Use `node[N].reset()` or `localnode.reset()` to reset only one of the nodes. The other nodes are not affected. The following programming example shows this type of reset operation with code that is run on node 1.

```
-- Reset node 1 only.
node[1].reset()
-- Reset the node you are connected to (in this case, node 1).
localnode.reset()
-- Reset node 4 only.
node[4].reset()
```

Terminating scripts on the TSP-Link system

You can terminate a script that is executing on a TSP-Link system.

To terminate an executing script and return all nodes to local control, send the following command:

```
abort
```

This dissolves the master/subordinate relationships between nodes.

You can also abort an executing script and turn off the source outputs on all source-measure units in the TSP-Link system. To do this, press the OUTPUT ON/OFF switch on any source-measure instrument in the system.

From the front panel, you can abort a script by pressing the TRIGGER key for a few seconds and selecting **Abort Trigger Model** from the dialog box that is displayed.

Triggering using TSP-Link synchronization lines

The Model DMM7510 has three synchronization lines that you can use for triggering, digital I/O, and to synchronize multiple instruments on a TSP-Link® network.

Using TSP-Link synchronization lines for digital I/O

Each synchronization line is an open-drain signal. When using the TSP-Link® synchronization lines for digital I/O, any node that sets the programmed line state to zero (0) causes all nodes to read 0 from the line state. This occurs regardless of the programmed line state of any other node. Refer to the table in the Digital I/O bit weighting (on page 3-53) topic for digital bit weight values.

Running simultaneous test scripts

Running test scripts simultaneously improves functional testing, provides higher throughput, and expands system flexibility. You can use TSP-Link and TSP scripting to run simultaneous test scripts. You can also manage the resources that are allocated to test scripts that are running simultaneously.

In addition, you can use the data queue to do real-time communication between nodes on the TSP-Link system.

To run test scripts simultaneously, you can set up your TSP-Link network in one of the following configurations:

* Multiple TSP-Link networks
* A single TSP-Link network with groups

Using groups to manage nodes on a TSP-Link system

TSP-Link groups allow each group to run a different test script simultaneously. This method requires one TSP-Link network and a single remote connection to the computer that is connected to the master node.

A group can consist of one or more nodes. You must assign group numbers to each node using remote commands. If you do not assign a node to a group, it defaults to group 0, which will always be grouped with the master node (regardless of the group to which the master node is assigned).

The following table shows an example of the functions of groups on a single TSP-Link network. Each group in this example runs a different test script than the other groups, which allows the system to run multiple tests simultaneously.

TSP-Link network group functions

Group number	Group members	Present function
0	Master node 1	Initiates and runs a test script on node 2 Initiates and runs a test script on node 6 In addition, the master node can execute scripts and process run commands
1	Group leader node 2	Runs the test script initiated by the master node Initiates remote operations on node 3 through node 5
	Node 3 through node 5	Performs remote operations initiated by node 2
2	Group leader node 6	Runs the test script initiated by the master node Initiates remote operations on node 7 through node n
	Node 7 through node n	Performs remote operations initiated by node 6

Master node overview

You can assign the master node to any group. You can also include other nodes in the group that includes the master. Note that any nodes that are set to group 0 are automatically included in the group that contains the master node, regardless of the group that is assigned to the master node.

The master node is always the node that coordinates activity on the TSP-Link network.

The master node:

- Is the only node that can use the `execute()` command on a remote node

- Cannot initiate remote operations on any node in a remote group if any node in that remote group is performing an overlapped operation (a command that continues to operate after the command that initiated it has finished running)

- Can execute the `waitcomplete()` command to wait for the group to which the master node belongs; to wait for another group; or to wait for all nodes on the TSP-Link network to complete overlapped operations (overlapped commands allow the execution of subsequent commands while device operations of the overlapped command are still in progress)

Group leader overview

Each group has a dynamic group leader. The last node in a group that performs any operation initiated by the master node is the group leader.

The group leader:

- Performs operations initiated by the master node
- Initiates remote operations on any node with the same group number
- Cannot initiate remote operations on any node with a different group number
- Can use the `waitcomplete()` command without a parameter to wait for all overlapped operations running on nodes in the same group

Assigning groups

Group numbers can range from zero (0) to 64. The default group number is 0. You can change the group number at any time. You can also add or remove a node to or from a group at any time.

Each time the power for a node is turned off, the group number for that node changes to 0.

The following example code dynamically assigns a node to a group:

```
-- Assign node 3 to group 1.
node[3].tsplink.group = 1
```

Running test scripts and programs on remote nodes

You can send the `execute()` command from the master node to initiate a test script and Lua code on a remote node. The `execute()` command places the remote node in the overlapped operation state. As a test script runs on the remote node, the master node continues to process other commands simultaneously.

Use the following code to send the `execute()` command for a remote node. The N parameter represents the node number that runs the test script (replace N with the node number).

To set the global variable "setpoint" on node N to 2.5:
```
node[N].execute("setpoint = 2.5")
```

The following code demonstrates how to run a test script that is defined on the local node. For this example, *scriptVar* is defined on the local node, which is the node that initiates the code to run on the remote node. The local node must be the master node.

To run scriptVar on node N:
```
node[N].execute(scriptVar.source)
```

The programming example below demonstrates how to run a test script that is defined on a remote node. For this example, *scriptVar* is defined on the remote node.

To run a script defined on the remote node:
```
node[N].execute("scriptVar()")
```

It is recommended that you copy large scripts to a remote node to improve system performance.

Coordinating overlapped operations in remote groups

All overlapped operations on all nodes in a group must have completed before the master node can send a command to the group. If you send a command to a node in a remote group when an overlapped operation is running on any node in that group, errors will occur.

You can execute the `waitcomplete()` command on the master node or group leader to wait for overlapped operations. The action of `waitcomplete()` depends on the parameters specified.

If you want to wait for completion of overlapped operations for:

- **All nodes in the local group:** Use `waitcomplete()` without a parameter from the master node or group leader.
- **A specific group:** Use `waitcomplete(N)` with a group number as the parameter from the master node. This option is not available for group leaders.
- **All nodes in the system:** Use `waitcomplete(0)` from the master node. This option is not available for group leaders.

For additional information, refer to [waitcomplete()](#) (on page 8-362).

The following code shows two examples of using the `waitcomplete()` command from the master node:

```
-- Wait for each node in group N to complete all overlapped operations.
waitcomplete(N)
-- Wait for all groups on the TSP-Link network to complete overlapped operations.
waitcomplete(0)
```

A group leader can issue the `waitcomplete()` command to wait for the local group to complete all overlapped operations.

The following code is an example of how to use the `waitcomplete()` command from a group leader:

```
-- Wait for all nodes in the local group to complete all overlapped operations.
waitcomplete()
```

Using the data queue for real-time communication

Nodes that are running test scripts at the same time can store data in the data queue for real-time communication. Each instrument has an internal data queue that uses the first-in, first-out (FIFO) structure to store data. You can use the data queue to post numeric values, strings, and tables.

Use the data queue commands to:

- Share data between test scripts running in parallel
- Access data from a remote group or a local node on a TSP-Link® network at any time

You cannot access the reading buffers or global variables from any node in a remote group while a node in that group is performing an overlapped operation. However, you can use the data queue to retrieve data from any node in a group that is performing an overlapped operation. In addition, the master node and the group leaders can use the data queue as a way to coordinate activities.

Tables in the data queue consume one entry. When a node stores a table in the data queue, a copy of the data in the table is made. When the data is retrieved from the data queue, a new table is created on the node that is retrieving the data. The new table contains a completely separate copy of the data in the original table, with no references to the original table or any subtables.

You can access data from the data queue even if a remote group or a node has overlapped operations in process. See the `dataqueue` commands in the TSP command reference (on page 8-1) for more information.

Remote TSP-Link commands

Commands that control and access the TSP-Link® synchronization port are summarized in the following table. See the TSP command reference (on page 8-1) for complete details on these commands.

Use the commands in the following table to perform basic steady-state digital I/O operations; for example, you can program the Model DMM7510 to read and write to a specific TSP-Link synchronization line or to the entire port.

TSP-Link commands

Command	Description
trigger.tsplinkin[N].clear() (on page 8-333)	Clears the event detector for a trigger
trigger.tsplinkin[N].edge (on page 8-334)	Indicates which trigger edge controls the trigger event detector for a trigger line
trigger.tsplinkin[N].overrun (on page 8-334)	Indicates if the event detector ignored an event while in the detected state
trigger.tsplinkin[N].wait() (on page 8-335)	Waits for a trigger
trigger.tsplinkout[N].assert() (on page 8-336)	Simulates the occurrence of the trigger and generates the corresponding trigger event
trigger.tsplinkout[N].logic (on page 8-336)	Defines the trigger output with output logic for a trigger line
trigger.tsplinkout[N].pulsewidth (on page 8-337)	Sets the length of time that the trigger line is asserted for output triggers
trigger.tsplinkout[N].release() (on page 8-338)	Releases a latched trigger on the given TSP-Link trigger line
trigger.tsplinkout[N].stimulus (on page 8-338)	Specifies the event that causes the synchronization line to assert a trigger
tsplink.group (on page 8-340)	The group number of the TSP-Link node
tsplink.initialize() (on page 8-341)	Initializes all instruments and enclosures in the TSP-Link system
tsplink.line[N].mode (on page 8-342)	Defines the trigger operation of a TSP-Link line as digital in or out or trigger in or out
tsplink.line[N].reset() (on page 8-343)	Resets some of the TSP-Link trigger attributes to their defaults
tsplink.line[N].state (on page 8-343)	Reads or writes the digital state of a TSP-Link synchronization line
tsplink.master (on page 8-344)	Reads the node number assigned to the master node
tsplink.node (on page 8-345)	Defines the node number
tsplink.readport() (on page 8-345)	Reads the TSP-Link synchronization lines as a digital I/O port
tsplink.state (on page 8-346)	Describes the TSP-Link online state
tsplink.writeport() (on page 8-346)	Writes to all TSP-Link synchronization lines as a digital I/O port

TSP-Link synchronization programming example

The programming example below illustrates how to set bit B1 of the TSP-Link digital I/O port high, and then read the entire port value:

```
tsplink.line[1].mode = tsplink.MODE_DIGITAL_OPEN_DRAIN
-- Set bit B1 high.
tsplink.line[1].state = 1
-- Read I/O port.
data = tsplink.readport()
print(data)
```

The output would be similar to:

```
7
```

To read bit B1 only:

```
-- To read bit B1 only
data = tsplink.line[1].state
print(data)
```

The output would be similar to:

```
tsplink.STATE_HIGH
```

Using Model DMM7510 TSP-Link commands with other TSP-Link products

If you are connecting the Model DMM7510 in a system with other TSP-Link products, be aware that some of the TSP-Link commands may be different. You can use the earlier versions of the commands, but be aware that they may not be supported in future versions of the product.

Commands that are the same in all TSP-Link products:

- `tsplink.group`
- `tsplink.master`
- `tsplink.node`
- `tsplink.readport()`
- `tsplink.state`
- `tsplink.writeport()`

Model DMM7510 TSP-Link command	Replaces this command in other TSP-Link products
`trigger.tsplinkin[N].clear()`	`tsplink.trigger[N].clear()`

`trigger.tsplinkin[N].edge`
`trigger.tsplinkout[N].logic`
`tsplink.line[N].mode`
	`tsplink.trigger[N].mode`

`trigger.tsplinkin[N].overrun`	`tsplink.trigger[N].overrun`
`trigger.tsplinkin[N].wait()`	`tsplink.trigger[N].wait()`
`trigger.tsplinkout[N].assert()`	`tsplink.trigger[N].assert()`
`trigger.tsplinkout[N].pulsewidth`	`tsplink.trigger[N].pulsewidth`
`trigger.tsplinkout[N].release()`	`tsplink.trigger[N].release()`
`trigger.tsplinkout[N].stimulus`	`tsplink.trigger[N].stimulus`
`tsplink.initialize()`	`tsplink.reset()`
`tsplink.line[N].reset()`	`tsplink.trigger[N].reset()`
`tsplink.line[N].state`	`tsplink.readbit()` `tsplink.writebit()`
Not applicable	`tsplink.writeprotect`

TSP-Net

TSP-Net provides a simple socket-like programming interface to Test Script Processor (TSP) enabled instruments. Using the TSP-Net library, the Model DMM7510 can control ethernet-enabled devices directly through its LAN port. This enables the Model DMM7510 to communicate directly with a device that is that is not TSP-enabled without the use of a controlling computer.

Using TSP-Net library methods, you can transfer string data to and from a remote instrument, transfer and format data into Lua variables, and clear input buffers. The TSP-Net library is only accessible using commands from a remote command interface when you are using the TSP command language.

While you can use TSP-Net commands to communicate with any ethernet-enabled instrument, specific TSP-Net commands exist for TSP-enabled instruments to allow for support of features unique to the TSP scripting engine. These features include script downloads, reading buffer access, wait completion, and handling of TSP scripting engine prompts.

Using TSP-Net commands with TSP-enabled instruments, a Model DMM7510 can download a script to another TSP-enabled instrument and have both instruments run scripts independently. The Model DMM7510 can read the data from the remote instrument and either manipulate the data or send the data to a different remote instrument on the LAN.

You can use TSP-Net to connect to a computer; you can use a script on the instrument to transfer data directly to your computer hard drive.

With TSP-Net, you can simultaneously connect to a maximum of 32 devices using standard TCP/IP networking techniques through the LAN port of the Model DMM7510.

Using TSP-Net with any ethernet-enabled instrument

<div style="border:1px solid #000">

NOTE

Refer to TSP command reference (on page 8-1) for details about the commands presented in this section.

</div>

The Model DMM7510 LAN port is auto-sensing (Auto-MDIX), so you can use either a LAN crossover cable or a LAN straight-through cable to connect directly from the Model DMM7510 to an ethernet device or to a hub.

To set up communication to a remote ethernet-enabled instrument that is TSP® enabled:

1. Send the following command to configure TSP-Net to send an abort command when a connection to a TSP instrument is established:

```
tspnet.tsp.abortonconnect = 1
```

    If the scripts are allowed to run, the connection is made, but the remote instrument may be busy.

2. Send the command:

```
connectionID = tspnet.connect(ipAddress)
```

    Where:

    *   *connectionID* is the connection ID that will be used as a handle in all other TSP-Net function calls.

    *   *ipAddress* is the IP address of the remote instrument.

See tspnet.connect() (on page 8-348) for additional detail.

To set up communication to a remote ethernet-enabled device that is not TSP enabled:

Send the command:

```
connectionID = tspnet.connect(ipAddress, portNumber, initString)
```

Where:

*   *connectionID* is the connection ID that will be used as a handle in all other `tspnet` function calls.

*   *ipAddress* is the IP address of the remote device.

*   *portNumber* is the port number of the remote device.

*   *initString* is the initialization string that is to be sent to *ipAddress*.

See tspnet.connect() (on page 8-348) for additional detail.

To communicate to a remote ethernet device from the Model DMM7510:

1. Connect to the remote device using one of the above procedures. If the Model DMM7510 cannot make a connection to the remote device, it generates a timeout event. Use `tspnet.timeout` to set the timeout value. The default timeout value is 20 s.

2. Use `tspnet.write()` or `tspnet.execute()` to send strings to a remote device. If you use:

    *   `tspnet.write()`: Strings are sent to the device exactly as indicated, and you must supply any needed termination characters.

    *   `tspnet.execute()`: The Model DMM7510 appends termination characters to all strings that are sent. Use `tspnet.termination()` to specify the termination character.

3. To retrieve responses from the remote instrument, use `tspnet.read()`. The Model DMM7510 suspends operation until the remote device responds or a timeout event is generated. To check if data is available from the remote instrument, use `tspnet.readavailable()`.

4. Disconnect from the remote device using the `tspnet.disconnect()` function. Terminate all remote connections using `tspnet.reset()`.

Example script

The following example demonstrates how to connect to a remote device that is not TSP® enabled, and send and receive data from this device:

```
-- Disconnect all existing TSP-Net connections.
tspnet.reset()
-- Set tspnet timeout to 5 s.
tspnet.timeout = 5
-- Establish connection to another device with IP address 192.168.1.51
-- at port 1394.
id_instr = tspnet.connect("192.168.1.51", 1394, "*rst\r\n")
-- Print the device ID from connect string.
print("ID is: ", id_instr)
-- Set the termination character to CRLF. You must do this
-- for each connection after the connection has been made.
tspnet.termination(id_instr, tspnet.TERM_CRLF)
-- Send the command string to the connected device.
tspnet.write(id_instr, "*idn?" .. "\r\n")
-- Read the data available, then print it.
print("instrument write/read returns: ", tspnet.read(id_instr))
-- Disconnect all existing TSP-Net sessions.
tspnet.reset()
```

Remote instrument events

If the Model DMM7510 is connected to a TSP-enabled instrument through TSP-Net, all events that occur on the remote instrument are transferred to the event log of the Model DMM7510. The Model DMM7510 indicates events from the remote instrument by prefacing these events with "Remote Error." For example, if the remote instrument generates event code 4909, "Reading buffer not found within device," the Model DMM7510 generates the string "Remote Error: (4909) Reading buffer not found within device."

TSP-Net instrument commands: General device control

The following instrument commands provide general device control:

tspnet.clear() (on page 8-347)
tspnet.connect() (on page 8-348)
tspnet.disconnect() (on page 8-349)
tspnet.execute() (on page 8-349)
tspnet.idn() (on page 8-350)
tspnet.read() (on page 8-351)
tspnet.readavailable() (on page 8-352)
tspnet.reset() (on page 8-353)
tspnet.termination() (on page 8-353)
tspnet.timeout (on page 8-354)
tspnet.write() (on page 8-357)

TSP-Net instrument commands: TSP-enabled device control

The following instrument commands provide TSP-enabled device control:

tspnet.tsp.abort() (on page 8-354)
tspnet.tsp.abortonconnect (on page 8-355)
tspnet.tsp.rbtablecopy() (on page 8-356)
tspnet.tsp.runscript() (on page 8-357)

Example: Using tspnet commands

```
function telnetConnect(ipAddress, userName, password)
   -- Connect through Telnet to a computer.
   id = tspnet.connect(ipAddress, 23, "")
   -- Read the title and login prompt from the computer.
   print(string.format("from computer--> (%s)", tspnet.read(id, "%n")))
   print(string.format("from computer--> (%s)", tspnet.read(id, "%s")))
   -- Send the login name.
   tspnet.write(id, userName .. "\r\n")
   -- Read the login echo and password prompt from the computer.
   print(string.format("from computer--> (%s)", tspnet.read(id, "%s")))
   -- Send the password information.
   tspnet.write(id, password .. "\r\n")
   -- Read the telnet banner from the computer.
   print(string.format("from computer--> (%s)", tspnet.read(id, "%n")))
   print(string.format("from computer--> (%s)", tspnet.read(id, "%n")))
   print(string.format("from computer--> (%s)", tspnet.read(id, "%n")))
   print(string.format("from computer--> (%s)", tspnet.read(id, "%n")))
end

function test_tspnet()
   tspnet.reset()
   -- Connect to a computer using Telnet.
   telnetConnect("192.0.2.1", "my_username", "my_password")
   -- Read the prompt back from the computer.
   print(string.format("from computer--> (%s)", tspnet.read(id, "%n")))
   -- Change directory and read the prompt back from the computer.
   tspnet.write(id, "cd c:\\\r\n")
   print(string.format("from computer--> (%s)", tspnet.read(id, "%s")))
   -- Make a directory and read the prompt back from the computer.
   tspnet.write(id, "mkdir TEST_TSP\r\n")
   print(string.format("from computer--> (%s)", tspnet.read(id, "%s")))
   -- Change to the newly created directory.
   tspnet.write(id, "cd c:\\TEST_TSP\r\n")
   print(string.format("from computer--> (%s)", tspnet.read(id, "%s")))
   -- if you have data print it to the file.
   -- 11.2 is an example of data collected.
   cmd = "echo " .. string.format("%g", 11.2) .. " >> datafile.dat\r\n"
   tspnet.write(id, cmd)
   print(string.format("from computer--> (%s)", tspnet.read(id, "%s")))
   tspnet.disconnect(id)
end
test_tspnet()
```

Measure considerations

Line cycle synchronization

Using line synchronization helps increase common-mode and normal-mode noise rejection. When line cycle synchronization is enabled, measurements are initiated at the first positive-going zero crossing of the power line cycle after the trigger.

Line cycle synchronization only applies to the following functions: Voltage, temperature, continuity, resistance, and DC voltage ratio.

You can enable line synchronization for NPLC measurements, which increases the normal-mode rejection ratio (NMRR) and common-mode rejection ratio (CMRR).

Using aperture or NPLCs to adjust speed and accuracy

You can adjust the amount of time that the input signal is measured. Adjustments to the amount of time affect the usable measurement resolution, the amount of reading noise, and the reading rate of the instrument.

> ## NOTE
> This discusses aperture for the measure functions. For information regarding aperture for the digitize functions, refer to Digitize functions (on page 2-124).

Depending on the function, you can set the time as an aperture or number of power line cycles (NPLCs).

When you set the time as an aperture, you set it as a number of seconds.

When you set the time in relation to NPLCs, you set it as the number of power line cycles that should occur during the measurement. Each power line cycle for 60 Hz is 16.67 ms (1/60); for 50 Hz, it is 20 ms (1/50).

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

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

The Model DMM7510 has a parabola-like shape for its speed versus noise characteristics. The Model DMM7510 is optimized for the 1 PLC to 5 PLC reading rate. At these rates (lowest noise region in graph), the Model DMM7510 will make corrections for its own internal drift and will still be fast enough to settle a step response of less than 100 ms.

Figure 129: Speed compared to noise characteristics
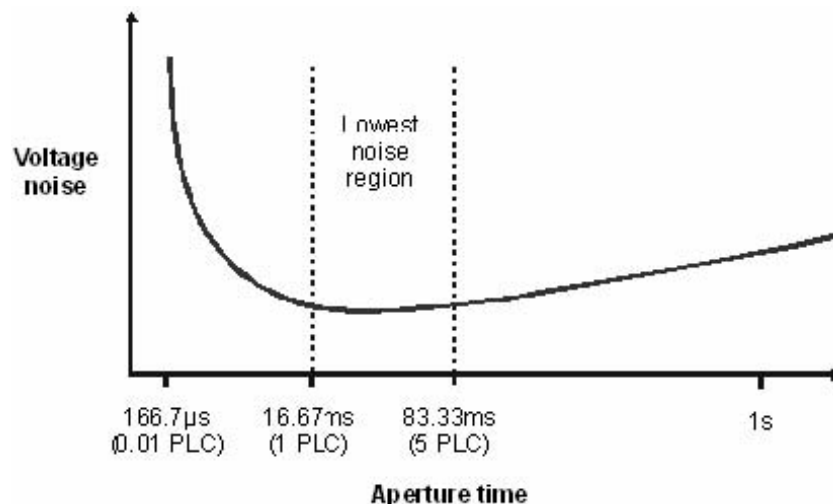
Rate cannot be set for continuity, capacitance, frequency, or period.

<div style="text-align:center">NOTE</div>

The Model DMM7510 uses internal references to calculate an accurate and stable reading. When the NPLC setting is changed, each reference is automatically updated to the new NPLC setting before a reading is generated. Therefore, frequent NPLC setting changes can result in slower measurement speed.

This setting also affects the normal mode rejection ratio (NMRR) and common mode rejection ratio (CMRR). Normal mode noise is the noise signal between the HI and LO terminals; common-mode noise is the noise signal between LO and chassis ground. See the Model DMM7510 specification for NMRR and CMRR values at different PLC settings.

If you change the aperture or NPLCs, you may want to adjust the displayed digits to reflect the change in usable digits. Refer to Setting the number of displayed digits (on page 2-52).

For functions that can accept either an aperture or an NPLC value, changing the value of one changes the value for the other. For example, if you set an aperture of 0.035, then set an NPLC value of 2, the aperture value is changed to 0.033333333.

To set NPLC using the front panel:
1. Press the **FUNCTION** key.
2. Select the measure function.
3. Press the **MENU** key.
4. Under Measure, select **Settings**.
5. Select **Integration Rate**. If the function allows both NPLC or aperture settings, the Integration Rate dialog box is displayed. Otherwise, a number pad is displayed.
6. If the Integration Rate dialog box is displayed, set the Unit to be **NPLC** or **Aperture**.
7. For NPLC or Aperture, enter the value.
8. Select **OK**.

DMM resistance measurement methods

The method that the Model DMM7510 uses to measure resistance depends on the resistance range. For resistance ranges from 1 Ω to 1 MΩ , the Model DMM7510 uses the constant-current method to measure resistance. For resistance ranges from 10 MΩ to 100 GΩ, the ratiometric method is used.

When the constant-current method is used, the Model DMM7510 sources a constant current (I) to the device under test and measures the voltage (V). Resistance (R) is then calculated and displayed using the known current and measured voltage (R = V/I).

When the ratiometric method is used, test current is generated by a 6.9 V reference through a 10 MΩ reference resistance (R_{REF}).

Constant-current source method

For the 1 Ω to 1 MΩ ranges, the Model DMM7510 uses the constant-current method to measure resistance. This method sources a constant current (I_{SOUR}) to the device under test (DUT) and measures the voltage (V_{MEAS}). Resistance (R_{DUT}) is then calculated and displayed using the known current and measured voltage.

Simple schematics of the 2-wire and 4-wire constant-current methods are shown below. The test current sourced to the DUT depends on the selected measurement range. For example, for the 100 Ω range, the test current is 1 mA. Because the voltmeter of the Model DMM7510 has high input impedance (>10 GΩ), virtually all the test current (1 mA) flows through the DUT. For a DUT that is ≤1 kΩ, 4-wire ohm measurements should be used as shown. Because the voltage is measured at the DUT, voltage drop in the test leads is eliminated (this voltage could be significant when measuring a low-ohm DUT).

Figure 130: Constant-current resistance measurement method

$$R_{DUT} = \frac{V_{MEAS}}{I_{SOUR}}$$

$$R_{DUT} = \frac{V_{MEAS}}{I_{SOUR}}$$
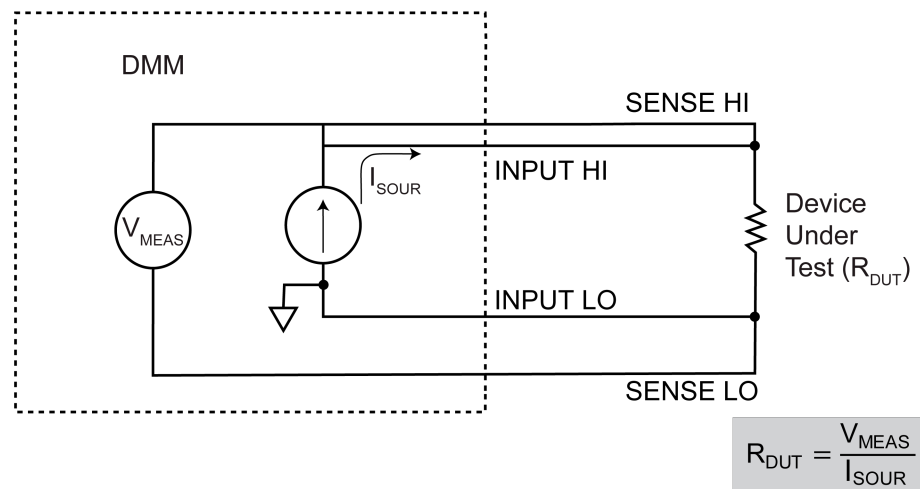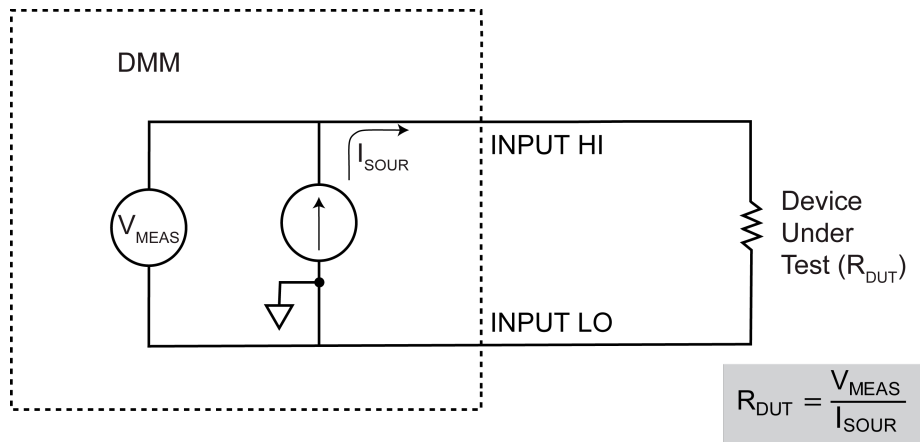
Ratiometric method

For the 10 MΩ through 1 GΩ ranges, the ratiometric method is used to measure resistance. Test current for this method is generated by a 6.9 V voltage source through a 10 MΩ reference resistance (R_{REF}), as shown in the figure below.

Basic circuit theory dictates that I_{REF} is equal to the I_{DUT}. Because the voltmeter of the Model DMM7510 (V_{MEAS}) has high input impedance (>10 GΩ), current through the voltmeter branch is insignificant and can be discounted. Therefore, as shown in the following figures, $I_{REF} = I_{DUT}$.

Figure 131: 2-wire ratiometric resistance measurement method schematic
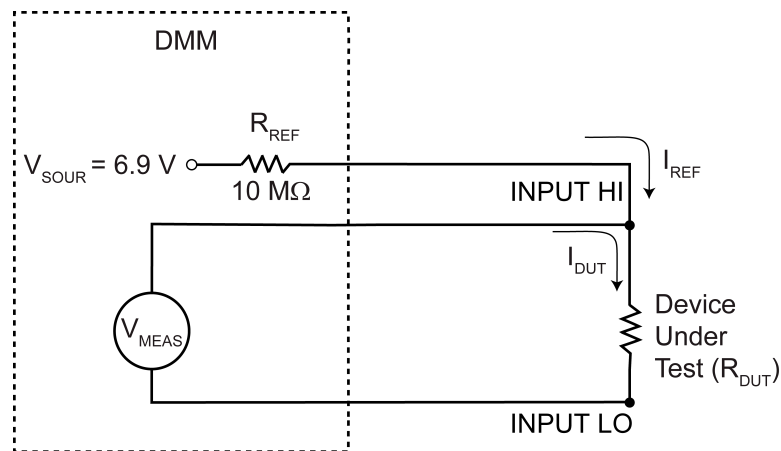
Equation 1:

$$I_{REF} = I_{DUT}$$

$$\frac{V_{SOUR} - V_{MEAS}}{R_{REF}} = \frac{V_{MEAS}}{R_{DUT}}$$

$$R_{DUT} = \frac{V_{MEAS}}{V_{SOUR} - V_{MEAS}} \times R_{REF}$$

Equation 2:

For R_{DUT} of approximately 0 ohms

$$I_{REF} = \frac{V_{SOUR}}{R_{REF}}$$

R_{DUT} example:

$$R_{DUT} \approx 10 \text{ ohms}$$

$$I_{REF} = \frac{V_{SOUR}}{R_{REF} + R_{DUT}}$$

$$I_{REF} = \frac{V_{SOUR}}{2R_{REF}}$$

Because I = V/R, Equation 1 is modified using the V/R equivalents in place of I_{REF} and I_{DUT}. Therefore:

$$I_{SOUR} = (V_{MEAS} / R_{REF}) + (V_{MEAS} / R_{DUT})$$

Note that V_{MEAS} is measured by the Model DMM7510. With V_{MEAS}, I_{SOUR}, R_{REF} known, the Model DMM7510 calculates the resistance of the DUT and displays the result. R_{REF} is learned during calibration and V_{SOUR} is routinely self-calibrated when autozero is enabled.

As shown, the 4-wire ohm function can also be used to measure ohms for the 10 MΩ and 100 MΩ ranges. To minimize the effects of charge injection when autozero is enabled, the 10 MΩ to 100 MΩ is actually a 3-wire ohm measurement. SENSE HI is not used (it can be left open). The measurement method is similar to the ratiometric method for 2-wire ohms, but it performs an extra voltage measurement (V_{LEAD}) to compensate for voltage drop in the input test leads.

Note that V_{MEAS} includes the voltage drops of the input test leads (Input HI and Input LO). Therefore, the actual voltage drop across the DUT is V_{MEAS} minus the two voltage drops in the test leads. Because matched inputs are used, the voltage drop is 2 x V_{LEAD}. Therefore:

$$V_{DUT} = V_{MEAS} - 2(V_{LEAD})$$

Figure 132: 4-wire ratiometric resistance measurement method
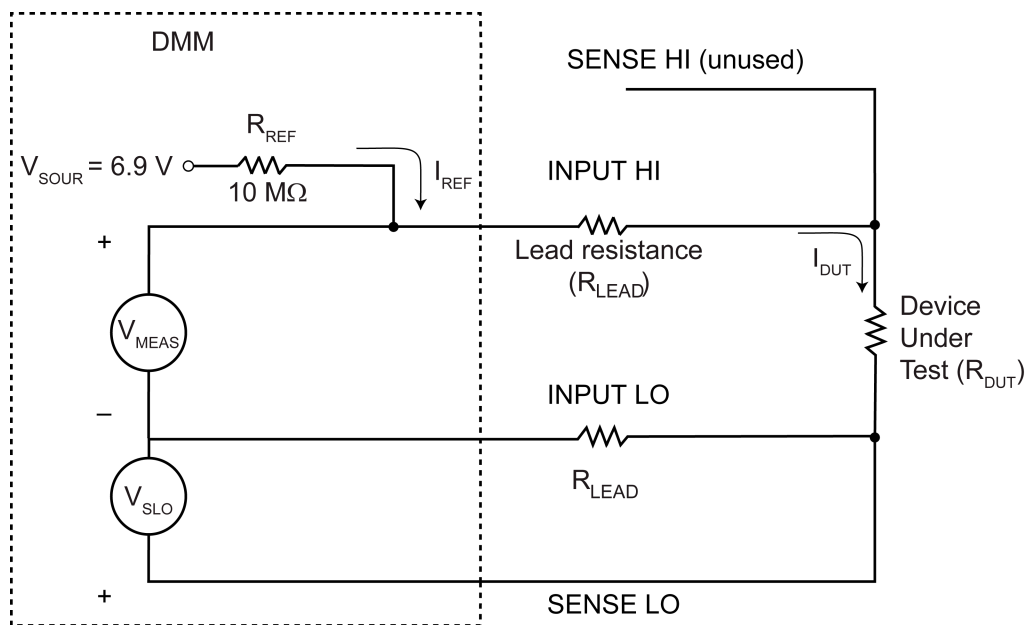
Low-level voltage measurement considerations

Low-level voltage measurements can be adversely affected by noise or other unwanted signals that can make it difficult to get accurate voltage readings. Some of the phenomena that can cause unwanted noise include thermoelectric effects (thermocouple action), source resistance noise, magnetic fields, and radio frequency interference. The following paragraphs discuss the most important of these effects and ways to minimize them.

NOTE

For comprehensive information on low-level measurements, see the *Low Level Measurements Handbook*, which is available from Keithley Instruments.

Thermoelectric potentials

Thermoelectric potentials, or thermoelectric EMFs, are the most common source of errors in low-voltage measurements. These small electric potentials are generated when different parts of the circuit are at different temperatures and when conductors made of dissimilar metals are joined together.

Thermoelectric EMFs can cause the following conditions:

* Instability or zero offset is much higher than expected.
* The reading is sensitive to and responds to temperature changes. This effect can be demonstrated by touching the circuit, by placing a heat source near the circuit, or by a regular pattern of instability (for example, corresponding to changes in sunlight or the activation of heating and air conditioning systems).

 The following paragraphs discuss how thermoelectric potentials are generated and ways to minimize their effects.

Thermoelectric coefficients

The table below shows the magnitude of thermoelectric EMFs that are generated for different materials.

Material thermoelectric coefficients

Material	Thermoelectric potential
Copper-to-Copper	0.2 μV/°C
Copper-to-Silver	0.3 μV/°C
Copper-to-Gold	0.3 μV/°C
Copper-to-Cadmium/Tin	0.3 μV/°C
Copper-to-Lead/Tin	1 to 3 μV/°C
Copper-to-Kovar	40 to 75 μV/°C
Copper-to-Silicon	400 μV/°C
Copper-to-Copper Oxide	1000 μV/°C

Thermoelectric EMF generation

The figure below shows how thermoelectric EMFs are generated.

The test leads are made of material A, while the source under test is material B. The temperatures between the junctions are shown as T_1 and T_2.

Figure 133: Thermoelectric EMF generation

Digital multimeter
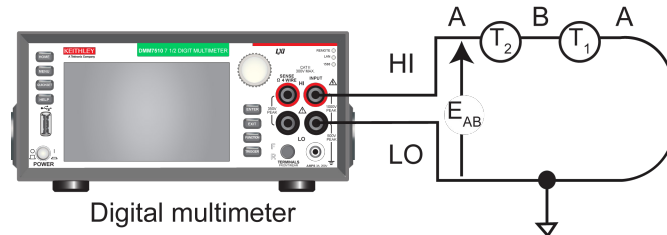
To calculate the thermoelectric EMFs that are generated:

$E_{AB} = Q_{AB} (T_1 - T_2)$

where:

- E_{AB} is the generated thermoelectric EMF
- Q_{AB} is the thermoelectric coefficient of material A with respect to material B ($\mu V/°C$)
- T1 is the temperature of the B junction (°C or K)
- T2 is the temperature of the A junction (°C or K)

A typical test setup has several copper-to-copper junctions. Each junction can have a thermoelectric coefficient as high as 0.2 $\mu V/°C$. Since the two materials frequently have a several degree temperature differential, thermoelectric EMFs of several microvolts can be generated even if reasonable precautions are taken.

Minimizing thermoelectric EMFs

To minimize thermoelectric EMF generation:

- Construct circuits that use the same material for all conductors. For example, connections made by crimping copper sleeves or lugs on copper wires result in copper-to-copper junctions, which generate minimal thermoelectric EMFs.

- Keep connections clean and free of oxides.

- Use low-thermoelectric cables and connections.

- Keep the two materials forming the junction at the same temperature.

- Keep the two junctions close together.

- Allow test equipment to warm up and reach thermal equilibrium in a constant ambient temperature.

- Keep all junctions away from air currents; in some cases, it may be necessary to thermally insulate sensitive junctions to minimize temperature variations.

- When making a copper-to-copper connection, apply sufficient pressure to ensure the connection is gas tight to prevent future oxidation.

- In some cases, you may need to connect the two thermal junctions together with good thermal contact to a common heat sink. Unfortunately, most good electrical insulators are poor heat conductors. In cases where low thermal conductivity may be a problem, you can use special insulators that combine high electrical insulating properties with high thermal conductivity. Some examples of these materials include hard anodized aluminum, sapphire, and diamond.

Using relative offset to minimize thermoelectric EMFs

Some systems may still have residual thermoelectric offsets after following the guidelines in Minimizing thermoelectric EMFs (on page 4-9). If the offsets are relatively constant, you can use the relative offset feature in the Model DMM7510 to cancel them. Refer to Relative offset (on page 3-5) for information.

Magnetic fields

When a conductor loop cuts through magnetic lines of force, a very small current is generated. This phenomenon can cause unwanted signals to occur in the test leads of a test system. If the conductor has sufficient length or cross-sectional area, even weak magnetic fields can create signals that affect low-level measurements.

To reduce these effects:

- Reduce the lengths of the connecting cables.

- Minimize the exposed circuit area.

- Change the orientation of the leads or cables.

In extreme cases, you may require magnetic shielding. Special metal with high permeability at low flux densities (such as mu metal) is effective at reducing these effects.

Even when the conductor is stationary, you may have problems with magnetically-induced signals. Fields can be produced by sources such as the AC power line voltage. Large inductors, such as power transformers, can generate substantial magnetic fields. Keep the Model DMM7510 voltage source and connecting cables away from these potential noise sources.

Radio frequency interference

Radio Frequency Interference (RFI) is a general term used to describe electromagnetic interference over a wide range of frequencies across the spectrum. RFI create problems at low signal levels, but it can also affect measurements at high levels if the fields are of sufficient magnitude.

RFI can be caused by steady-state sources, such as radio or TV signals, or some types of electronic equipment, such as microprocessors and high speed digital circuits. It can also result from impulse sources, as in the case of arcing in high-voltage environments. The effect on the measurement can be considerable if enough of the unwanted signal is present.

You can minimize RFI in several ways:

- Keep the Model DMM7510 voltage source and signal leads away from RFI sources.
- Shield instrument, signal leads, sources, and other measuring instruments.
- In extreme cases, a specially-constructed screen room may be required to sufficiently attenuate the RFI signal.

In some situations, the Model DMM7510 digital filter may help to reduce RFI effects. In some cases, additional external filtering may also be required. However, filtering may have detrimental effects, such as increased settling time on the signal.

Shielding

AC voltages that are extremely large compared to the DC signal to be measured may produce an erroneous output. Therefore, to minimize AC interference, the circuit should be shielded, with the shield connected to the Model DMM7510 input low (particularly for low-level sources). Improper shielding can cause the Model DMM7510 to behave in one or more of the following ways:

• Unexpected offset voltages

• Inconsistent readings between ranges

• Sudden shifts in reading

To minimize pick-up, keep the voltage source and the Model DMM7510 away from strong AC magnetic sources. The voltage induced due to magnetic flux is proportional to the area of the loop formed by the input leads. Therefore, minimize the loop area of the input leads and connect each signal at only one point.

To minimize noise, you may need a closed metal shield that surrounds the source. This shield should be connected to input LO in most cases. In some situations, you may get better noise performance with the shield connected to chassis ground.

Figure 134: Model DMM7510 rear panel noise shield connections
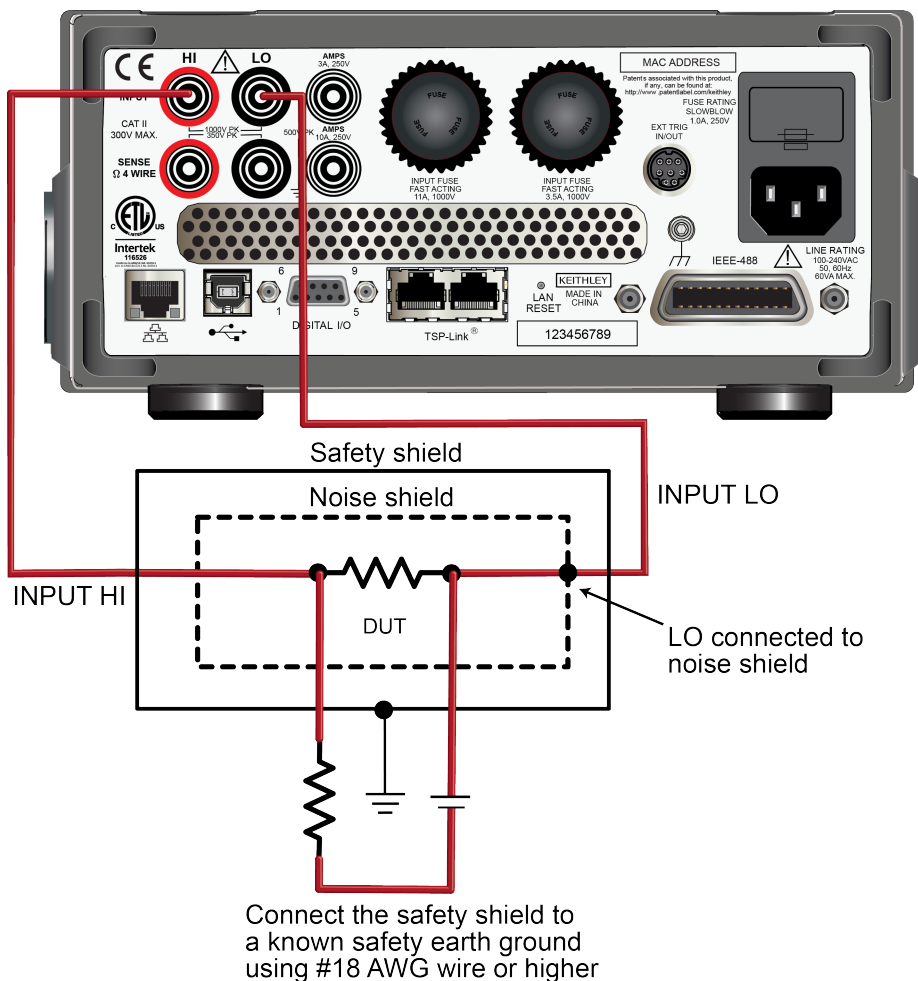
⚠ WARNING

INPUT and SENSE LO are not internally connected to the chassis and cannot be allowed to float above chassis ground more than the values shown on the front panel.

Cable effects on dry-circuit ohms

Dry-circuit resistance applications include measurements where the voltage applied across the device under test DUT is limited to 20 mV to 30 mV. The limited voltage is an important parameter when measuring resistance of metal oxides on connectors, relays, or switch devices contacts during environmental HALT and HASS testing.

The Model DMM7510 properly conditions the internal I_{test} current and voltage clamping signals to maintain less than 30 mV across the DUT.

Cable effects

A voltage this small can easily be generated during handling and connecting of the DUT. Therefore, you may need to protect the DUT during handling. One method for protecting the DUT is to short the DUT during handling, and then removing the short before testing.

Figure 135: DMM shorted during handling
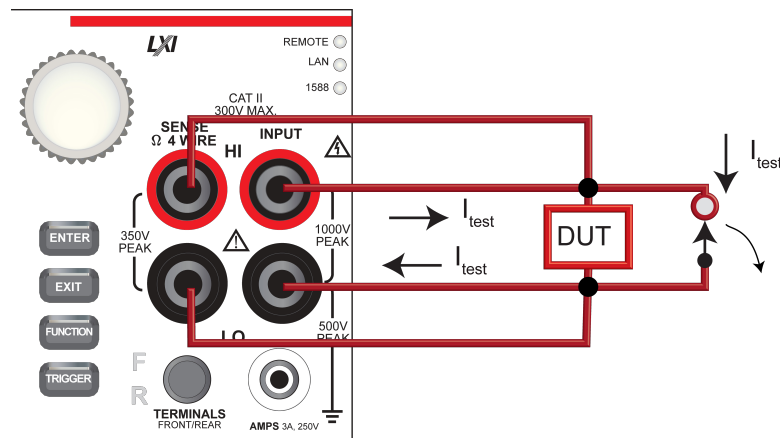
As long as the dry-circuit limits the voltage between HI and LO, this should be a good solution. However, when the short is in place, there is test current flowing through the cable that connects the HI and LO of the DMM to the DUT. When the short is removed, the energy stored in this cable can force a voltage across the DUT that far exceeds the 20 mV to 30 mV limitation.

If the HI and LO connection is modeled as a transmission line, when the short is opened, the maximum voltage across the DUT is $V_{dut} = I_{test}R_c$. If I_{test} is 10 mA and R_c is 50 ohms, V_{dut} is 0.5 V, which is more than 30 times the limit. The voltage on the DUT oscillates as the signal bounces back and forth in the cable until losses absorbs the energy, as shown in the simulation plot below. The second plot shows the voltage between HI and LO during the same simulation.

Figure 136: HI and LO modeled as a transmission line
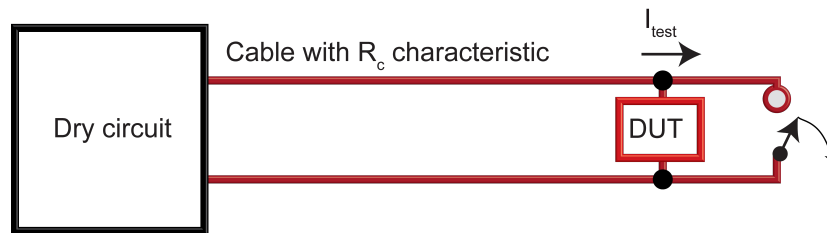
Figure 137: Voltage oscillation
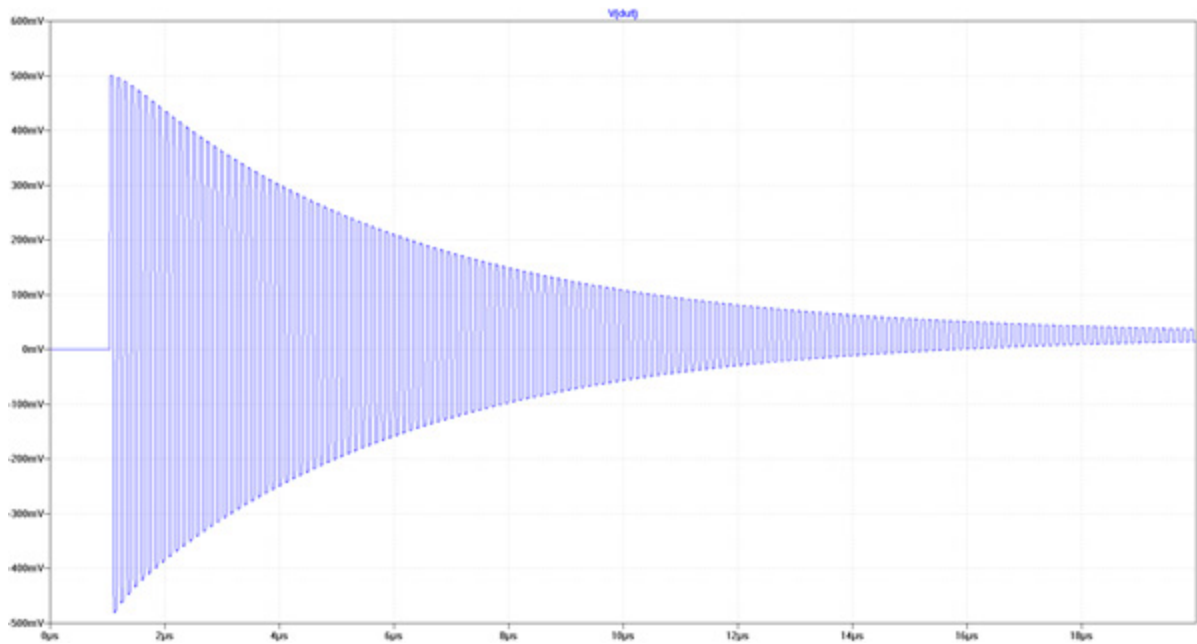
Figure 138: Voltage between HI and LO
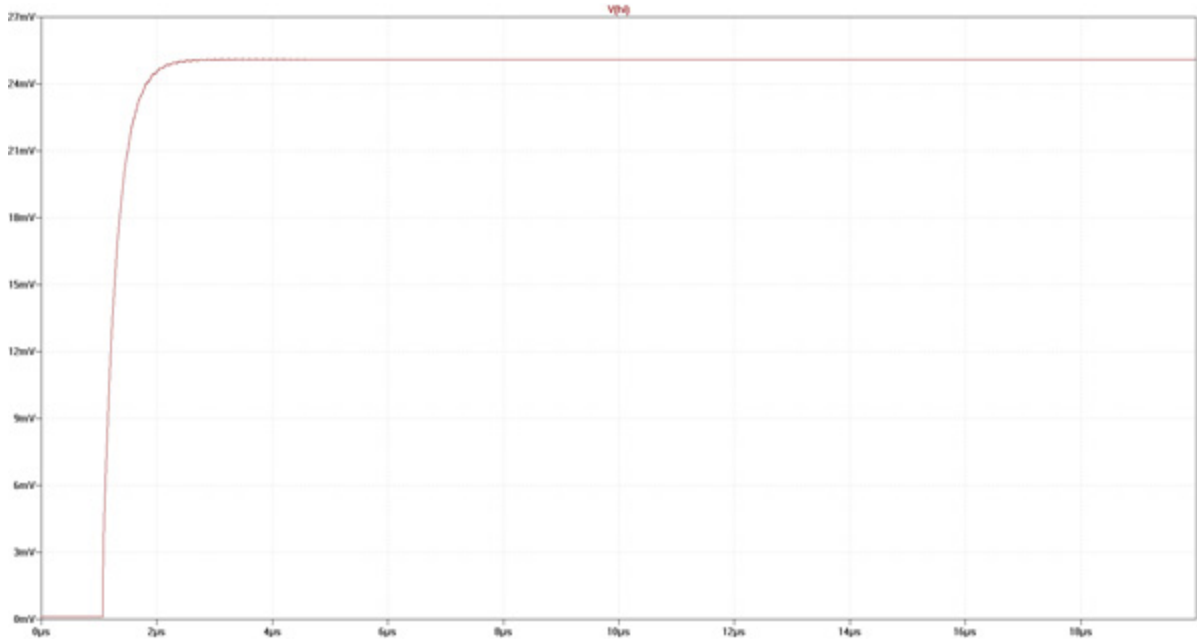
Solutions

Possible solutions to address cable effects:

1. Add a second switch to engage the test current after the short is removed, as shown in the following figure.

Figure 139: Cable effects solution
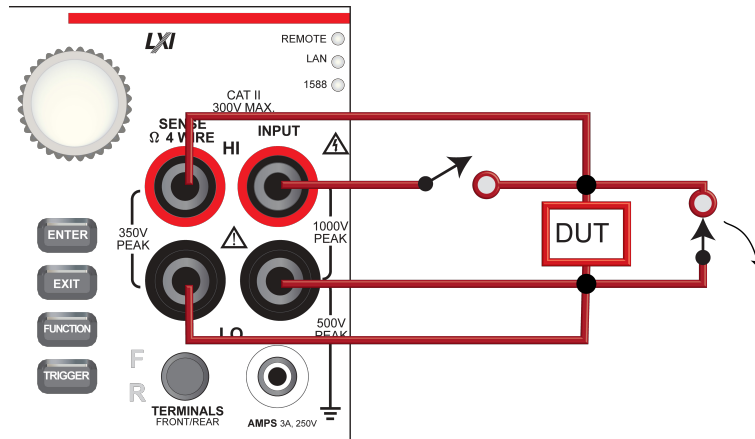
2. Set the DMM to a higher dry-circuit ohms range as the 10 KΩ range with 5 μA of I_{test} current. The cable effects are directly reduced with reduced I_{test} current.

3. For the cable that connects HI and LO to the DUT, choose a cable with a very low characteristic impedance so that the maximum DUT voltage is small.

4. Terminate the end of the cable with an RC snubber that is less than 1 Ω in series with a parallel 10 μF and 1 MΩ, which maintains low voltage and a single damped response.

NOTE

With offset compensation enabled and reading triggering halted, the idle test current of all dry-circuit ranges is less than or equal to 10 μA. When the Model DMM7510 is in either of these setups, the shorting switch across the DUT can be safely opened with minimal cable energy storage issues.

Offset-compensated ohm calculations

The presence of thermoelectric EMFs (V_{EMF}) can adversely affect low-resistance measurement accuracy. To overcome these offset voltages, you can use offset-compensated ohms if you are making 4-wire resistance measurements on ranges up to 100 kΩ.

NOTE

Instrument operations, including offset-compensated ohms, are performed on the input signal in a sequential manner.

For a normal resistance measurement, the Model DMM7510 sources a current (I) and measures the voltage (V). The resistance (R) is then calculated as (R=V/I) and the reading is displayed.

For offset-compensated ohms, two measurements are performed: one normal resistance measurement, and one using the lowest current source setting.

The offset-compensated ohms reading is then calculated as follows:

$$\text{Offset-compensated } \Omega = \frac{\Delta V}{\Delta I}$$

where:

$$\Delta V = V_2 - V_1$$
$$\Delta I = I_2 - I_1$$

V_1 is the voltage measurement with the current source at its normal level.

V_2 is the voltage measurement using the lowest current source setting.

I_1 is the current measurement with the source set to a specific level.

I_2 is the current measurement with the source set to zero.

This 2-point measurement process and reading calculation eliminates the resistance contributed by the presence of V_{EMF}.

When the source is turned on, the output cycles between the programmed value and zero (0 A or 0 V) to derive the offset-compensated ohms measurement.

Order of operations

The measurements have filtering, relative offset values, math operations, and limit testing applied to them in a pre-determined order. The measurements that are displayed on the front panel and that are stored in the reading buffers represent the measurement with any selected operations applied to them.

These operations are applied to the measurement as shown in the following figure.

Figure 140: Model DMM7510 order of operations
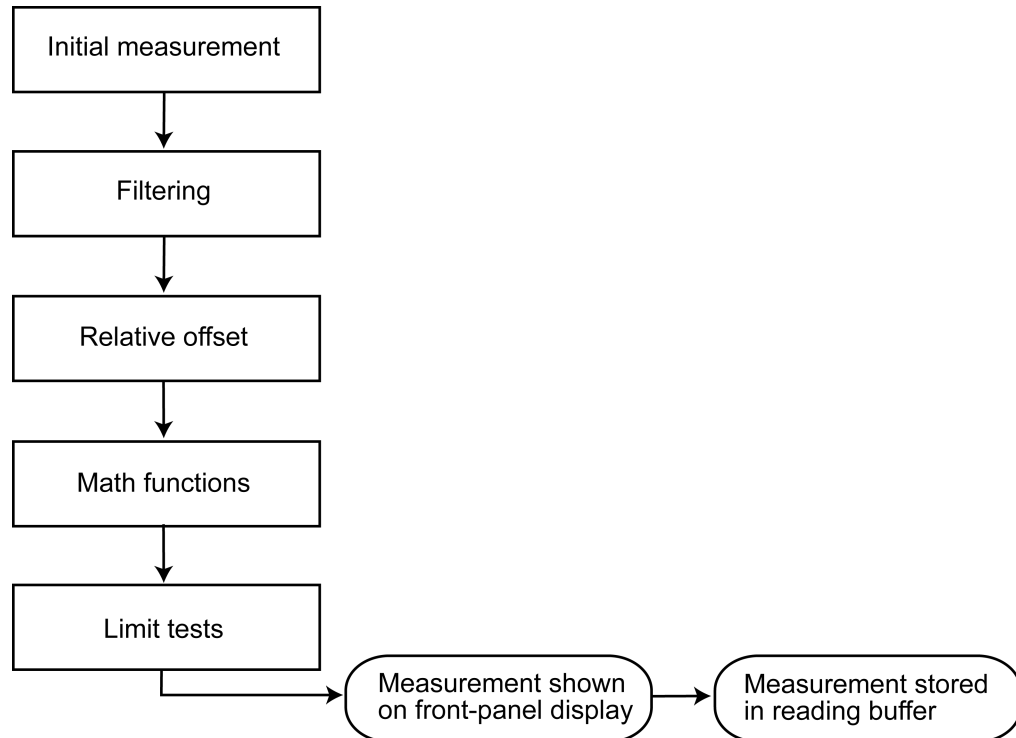
For more information on these operations, see the following topics:

- Filtering measurement data (on page 3-10)
- Relative offset (on page 3-5)
- Calculations that you can apply to measurements (on page 3-6)
- Limit testing and binning (on page 3-99)

Introduction to SCPI commands

In this section:

Introduction to SCPI

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

The following information describes some basic SCPI command information and how SCPI is used with the Model DMM7510 and presented in the Model DMM7510 documentation.

Command execution rules

Command execution rules are as follows:

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

Command messages

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

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

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

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

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

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

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

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

```
*RST
```

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

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

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

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

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

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

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

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

You can also send:

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

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

```
0.5;0
```

SCPI command programming notes

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

SCPI command formatting

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

SCPI command short and long forms

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

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

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

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

Optional command words

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

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

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

MINimum, MAXimum, and DEFault

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

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

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

Queries

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

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

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

This query returns the present setting.

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

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

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

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

SCPI parameters

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

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

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

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

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

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

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

Sending strings

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

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

Using the SCPI command reference

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

Figure 141: SCPI command description example

:EXAMple:COMMand:STATe

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

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

Usage

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

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

Details

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

Example

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

Also see

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

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

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

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

Command name and summary table

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

:EXAMple:COMMand:STATe

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

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	1 (ON)
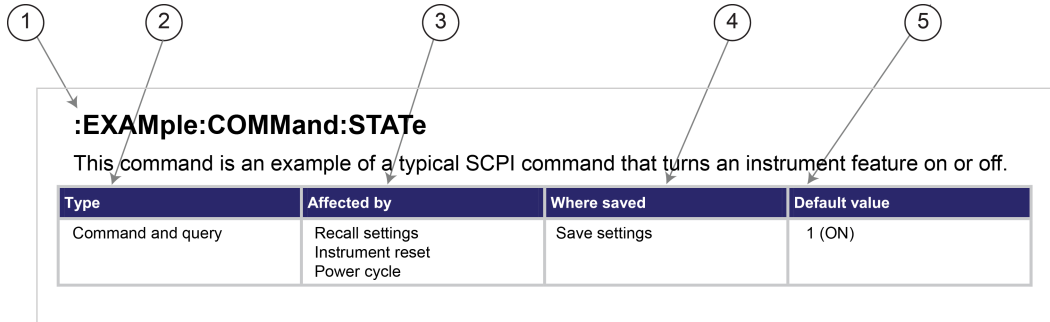
1   Instrument command name. Signals the beginning of the command description and is followed by a brief description of what the command does.

2   Type of command. Options are:

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

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

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

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

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

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

Command usage

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

Figure 142: SCPI command description usage identification
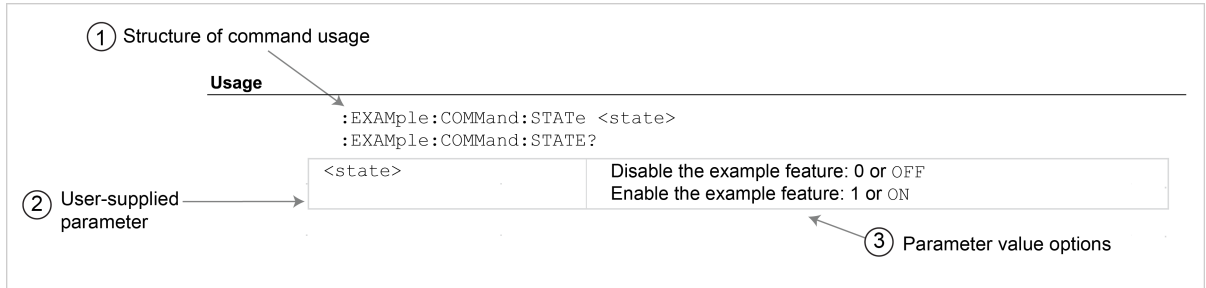
1   **Structure of command usage:** Shows how the parts of the command should be organized.
2   **User-supplied parameters:** Indicated by angle brackets (< >).

NOTE

Some commands have optional parameters. Optional parameters are presented on separate lines in the Usage section, presented in the required order with each valid permutation of optional parameters. For example:
```
:SYSTem:COMMunication:LAN:CONFigure AUTO
:SYSTem:COMMunication:LAN:CONFigure MANual, IPaddress
:SYSTem:COMMunication:LAN:CONFigure MANual, IPaddress, NETmask
:SYSTem:COMMunication:LAN:CONFigure MANual, IPaddress, NETmask, GATeway
:SYSTem:COMMunication:LAN:CONFigure?
```

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

Command details

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

Figure 143: Details section of command listing
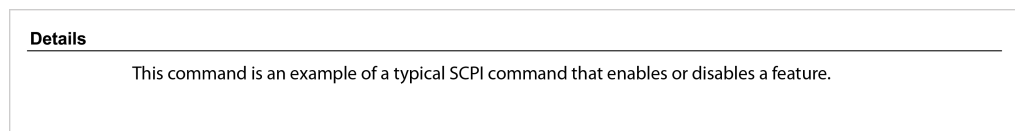
Example section

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

Figure 144: SCPI command description code examples

Example	
① Working code → `:EXAMple:COMMand:STATe ON` example	Turn the example feature on. ↑ ② Description of what code does
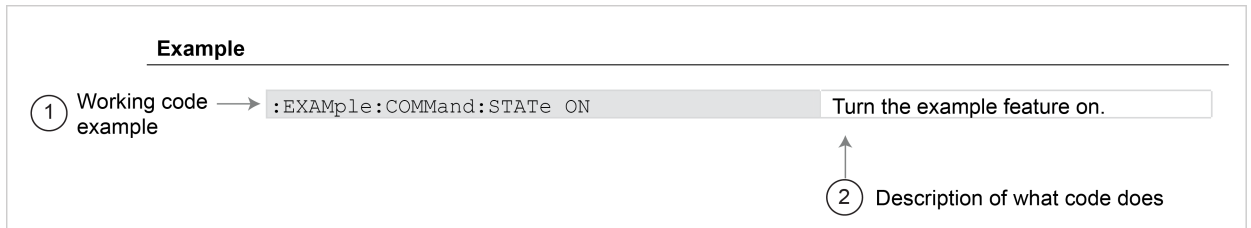
1 Example code that you can copy from this table and paste into your own application. Examples are generally shown using the short forms of the commands.

2 Description of the code and what it does. This may also contain the output of the code.

Related commands list

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

Figure 145: SCPI related commands list example

Also see
:EXAMple:UNIT[:IMMediate] (on page 6-99)

SCPI command reference

In this section:

:FETCh?

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

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

Usage

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

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

Details

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

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

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

When specifying buffer elements, you can:

- Specify buffer elements in any order.
- Include up to 12 elements in a single list. You can repeat elements as long as the number of elements in the list is less than 12.
- Use a comma to delineate multiple elements for a data point.

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

Option	Description
DATE	The date when the data point was measured; not available for reading buffers that are set to the style compact
EXTRa	Returns an additional value (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds when the data point was measured
READing	The measurement reading
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
STATus	The status information associated with the measurement
TIME	The time when the data point was measured
TSTamp	The timestamp when the data point was measured
UNIT	The unit of measure of the measurement

Example

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

Also see

:MEASure?

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<bufferName>`	The name of the buffer where the reading is stored; if nothing is specified, the reading that is stored in `defbuffer1`
`<bufferElements>`	See **Details**

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

If you do not define the function parameter, the instrument uses the presently selected measure function. If a digitize function is presently selected, an error is generated.

When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

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

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

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

`:MEASure?` performs the same function as `READ?`.

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

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

When specifying buffer elements, you can:

- Specify buffer elements in any order.
- Include up to 12 elements in a single list. You can repeat elements as long as the number of elements in the list is less than 12.
- Use a comma to delineate multiple elements for a data point.

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

Option	Description
DATE	The date when the data point was measured; not available for reading buffers that are set to the style compact
EXTRa	Returns an additional value (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds when the data point was measured
READing	The measurement reading
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
STATus	The status information associated with the measurement
TIME	The time when the data point was measured
TSTamp	The timestamp when the data point was measured
UNIT	The unit of measure of the measurement

Example

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

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

Also see

:MEASure:DIGitize?

This command makes a digitize measurement, places it in a reading buffer, and returns the reading.

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

Usage

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

`<function>`	The function to use for the measurement: • Voltage: `VOLTage` • Current: `CURRent` If no function is defined, the presently selected one is used
`<bufferName>`	The name of the buffer where the reading is stored; if nothing is specified, the reading that is stored in `defbuffer1`
`<bufferElements>`	See **Details**

Details

This command makes a digitize measurement using the specified function and stores the reading in a reading buffer. Sending this command changes the measurement function to the one specified by `<function>`. This function remains selected after the measurement is complete.

If you do not define the function parameter, the instrument uses the presently selected function. If a measure function is presently selected, an error is generated.

When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

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

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

`:MEASure:DIGitize?` performs the same function as `READ:DIGitize?`.

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

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

When specifying buffer elements, you can:

- Specify buffer elements in any order.

- Include up to 12 elements in a single list. You can repeat elements as long as the number of elements in the list is less than 12.

- Use a comma to delineate multiple elements for a data point.

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

Option	Description
DATE	The date when the data point was measured; not available for reading buffers that are set to the style compact
EXTRa	Returns an additional value (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds when the data point was measured
READing	The measurement reading
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
STATus	The status information associated with the measurement
TIME	The time when the data point was measured
TSTamp	The timestamp when the data point was measured
UNIT	The unit of measure of the measurement

Example

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

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

Also see

:READ?

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

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

Usage

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

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

Details

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

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

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

When specifying buffer elements, you can:

- Specify buffer elements in any order.

- Include up to 12 elements in a single list. You can repeat elements as long as the number of elements in the list is less than 12.

- Use a comma to delineate multiple elements for a data point.

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

Option	Description
DATE	The date when the data point was measured; not available for reading buffers that are set to the style compact
EXTRa	Returns an additional value (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds when the data point was measured
READing	The measurement reading
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
STATus	The status information associated with the measurement
TIME	The time when the data point was measured
TSTamp	The timestamp when the data point was measured
UNIT	The unit of measure of the measurement

Example

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

Create a buffer named `voltMeasBuffer`.

Set the measurement function to voltage.

Set the count to 10.

Make the measurements and store them in the buffer `voltMeasBuffer`. Return the last reading as displayed on the front panel with the date, along with the unformatted reading.

Return all 10 readings from the reading buffer.

Example output is:

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

Also see

:FETCh? (on page 6-1)
[:SENSe[1]]:COUNt (on page 6-116)
:TRACe:DATA? (on page 6-147)
:TRACe:TRIGger (on page 6-164)

:READ:DIGitize?

This query makes a digitize measurement, places it in a reading buffer, and returns the latest reading.

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

Usage

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

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

Details

You must set the instrument to a digitize function before sending this command.

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

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

When specifying buffer elements, you can:

- Specify buffer elements in any order.

- Include up to 12 elements in a single list. You can repeat elements as long as the number of elements in the list is less than 12.

- Use a comma to delineate multiple elements for a data point.

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

Option	Description
DATE	The date when the data point was measured; not available for reading buffers that are set to the style compact
EXTRa	Returns an additional value (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds when the data point was measured
READing	The measurement reading
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
STATus	The status information associated with the measurement
TIME	The time when the data point was measured
TSTamp	The timestamp when the data point was measured
UNIT	The unit of measure of the measurement

Example

```
*RST
:TRACe:MAKE "voltDigBuffer", 10000
:DIG:FUNC "VOLTage"
:SENS:DIG:COUN 100
:READ:DIG? "voltDigBuffer", FORM, DATE, READ
:TRAC:DATA? 95,100, "voltDigBuffer"
```

Create a buffer named voltDigBuffer. Make a digitize measurement, store it in the buffer voltDigBuffer, and return the formatted readings, date, and reading buffer elements for the last reading stored in voltDigBuffer, then return readings 95 to 100.

Example output is:

```
+04.963 V,09/26/2014,4.962954E+00
4.961211E+00,4.961695E+00,4.961889E+00,4.961985E+00,4.962276E+00,4.962954E+00
```

Also see

:FETCh? (on page 6-1)
[:SENSe[1]]:DIGitize:COUNt (on page 6-117)
[:SENSe[1]]:DIGitize:FUNCtion[:ON] (on page 6-118)
:TRACe:DATA? (on page 6-147)
:TRACe:MAKE (on page 6-152)
:TRACe:TRIGger:DIGitize (on page 6-165)

*RCL

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

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

Usage

```
*RCL <n>
```

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

Details

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

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

Example

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

Also see

Saving setups (on page 2-148)
*SAV (on page 6-12)

*SAV

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

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

Usage

```
*SAV <n>
```

<n>	An integer from 0 to 4

Details

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

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

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

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

Example

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

Also see

Saving setups (on page 2-148)
*RCL (on page 6-11)

ACAL subsystem

Automatic calibration removes measurement errors that are caused by the performance drift on the components used in the DMM as a result of temperature and time.

:ACAL:COUNt?

This command returns the number of times automatic calibration has been run.

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

Usage

```
:ACAL:COUNt?
```

Details

The number of times that auto calibration has been run since the last factory calibration. The count restarts at 1 after a factory calibration.

Example

`ACAL:COUN?`	Returns the number of times auto calibration has been run. Example output: `15`

Also see

Auto calibration (on page 3-40)
:ACAL:RUN (on page 6-18)

:ACAL:LASTrun:TEMPerature:INTernal?

This command returns the internal temperature of the instrument when auto calibration was run.

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

Usage

```
:ACAL:LASTrun:TEMPerature:INTernal?
```

Details

The temperature is displayed in Celsius (°C).

The instrument updates the internal temperature value when the instrument refreshes autozero. If autozero is set to off or if autozero is not available for the selected function (such as capacitance, continuity, frequency or period), the internal temperature value is not updated.

Example

`ACAL:LAST:TEMP:INT?`	Returns the internal temperature of the instrument when auto calibration was last run. Example output: `63.167084`

Also see

:ACAL:RUN (on page 6-18)
Auto calibration (on page 3-40)

:ACAL:LASTrun:TEMPerature:DIFFerence?

This command returns the difference between the internal temperature and the temperature when auto calibration was last run.

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

Usage

```
:ACAL:LASTrun:TEMPerature:DIFFerence?
```

Details

The temperature is displayed in Celsius (°C).

The instrument updates the internal temperature value when the instrument refreshes autozero. If autozero is set to off or if autozero is not available for the selected function (such as capacitance, continuity, frequency or period), the internal temperature value is not updated.

Example

`ACAL:LAST:TEMP:DIFF?`	Returns the difference between the temperature of the instrument when auto calibration was last run and the present internal temperature. Example output: `4.5678`

Also see

:ACAL:LASTrun:TEMPerature:INTernal? (on page 6-14)
:ACAL:RUN (on page 6-18)
Auto calibration (on page 3-40)

:ACAL:LASTrun:TIME?

This command returns the date and time when auto calibration was last run.

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

Usage

```
:ACAL:LASTrun:TIME?
```

Details

The date and time is returned in the format:

`MM/DD/YYYY HH:MM:SS.NNNNNNNNN`

Where:

- `MM/DD/YYYY` is the month, date, and year

- `HH:MM:SS.NNNNNNNNN` is the hour, minute, second, and fractional second

Example

`ACAL:LAST:TIME?`	Returns the date and time when auto calibration was last run. Example output: `08/11/2014 16:30:26.745369595`

Also see

[:ACAL:RUN](on page 6-18)

:ACAL:NEXTrun:TIME?

This command returns the date and time when the next auto calibration is scheduled to be run.

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

Usage

```
:ACAL:NEXTrun:TIME?
```

Details

The date and time is returned in the format:
`MM/DD/YYYY HH:MM:SS.NNNNNNNNN`
Where:

- `MM/DD/YYYY` is the month, date, and year

- `HH:MM:SS.NNNNNNNNN` is the hour, minute, second, and fractional second

Example

`ACAL:NEXT:TIME?`	Returns date and time when the next auto calibration is scheduled to be run. Example output: `05/29/2014 17:11:17.000000000`

Also see

:ACAL:RUN (on page 6-18)
:ACAL:SCHedule (on page 6-19)

:ACAL:REVert

This command returns auto calibration constants to the previous constants.

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

Usage

```
:ACAL:REVert
```

Details

This command reverts the present set of auto calibration constants to the previous set of auto calibration constants.
The last run time and internal temperature are reverted to the previous values. The auto calibration count is not changed.

Example

`ACAL:REV`	Auto calibration values are reverted to the previous set of auto calibration constants.

Also see

:ACAL:RUN (on page 6-18)

:ACAL:RUN

This command immediately runs auto calibration and stores the constants.

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

Usage

```
:ACAL:RUN
```

Details

During auto calibration, a progress message is displayed on the front panel. At completion, an event message is generated.

If you have set up auto calibration to run at a scheduled interval, when you send the run command, the instrument adjusts the next scheduled auto calibration to be the next interval. For example, if auto calibration is scheduled to run every 7 days, but you run auto calibration on day 3, the next auto calibration will run 7 days after day 3.

Example

`ACAL:RUN`	Auto calibration starts running.

Also see

:ACAL:SCHedule (on page 6-19)
Auto calibration (on page 3-40)

:ACAL:SCHedule

This command sets how often auto calibration occurs or prompts you to run it.

Type	Affected by	Where saved	Default value
Command and query	Recall setup	Nonvolatile memory Saved setup	Run every 8 hours starting at midnight

Usage

```
:ACAL:SCHedule <action>, <interval>
:ACAL:SCHedule <action>, <interval>, <hour>
:ACAL:SCHedule NONE
:ACAL:SCHedule?
```

`<action>`	Determines when and if the instrument automatically runs auto calibration: • To run auto calibration at the scheduled time: `RUN` • To notify you that the auto calibration needs to be run at the scheduled time: `NOTIFY` • To turn off scheduling: `NONE`; no other parameters are needed if none is selected
`<interval>`	Determines how often auto calibration should be run or notification should occur: • Every 8 hours: `HOUR8` • Every 16 hours: `HOUR16` • Every day: `DAY1` • Every 7 days: `DAY7` • Every 14 days: `DAY14` • Every 30 days: `DAY30` • Every 90 days: `DAY90`
`<hour>`	Specify when the auto calibration should occur; specify in 24-hour time format (`0` to `23`; default is `0`); not available for the 8-hour or 16-hour interval

Details

Auto calibration does not start until actions that are active on the instrument are complete. For example, if a trigger model is running when auto calibration is scheduled to run, auto calibration does not start until the trigger model stops.

If the scheduled time for auto calibration occurs before the warmup period completes, the instrument will not start auto calibration. The instrument waits until the warmup period is complete before starting a scheduled auto calibration. A message is displayed when warmup is complete and auto calibration is going to run.

If the instrument is powered off when an auto calibration was scheduled, auto calibration is run as soon as the warmup period is complete when the instrument is powered on.

You can run auto calibration manually even if a scheduled auto calibration is set.

When auto calibration is scheduled to run at a scheduled interval, but it runs at another time, subsequent scheduled intervals are adjusted according to the actual auto calibration start time.

Example

`ACAL:SCH RUN, DAY1, 8` `ACAL:SCH?`	Sets auto calibration to run every day at 8 am and query to verify the settings. Output: `RUN;DAY1;8`

Also see

CALCulate subsystem

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

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<Y>`	Limit number: `1` or `2`
`<state>`	When the beeper sounds: • Never: `NONE` • On test failure: `FAIL` • On test pass: `PASS`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

The tone and length of beeper cannot be adjusted.

Example

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

Also see

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<Y>`	Limit number: 1 or 2
`<state>`	The auto clear setting: • Disable: OFF or 0 • Enable: ON or 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

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

Example

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

Also see

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<Y>`	Limit number: `1` or `2`

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

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

Example

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

Also see

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

This command queries the results of a limit test.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<Y>`	Limit number: `1` or `2`

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

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

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

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

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

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

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

Example

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

Also see

:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO (on page 6-21)
:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate] (on page 6-22)
:CALCulate2:<function>:LIMit<Y>:STATe (on page 6-26)
Limit testing and binning (on page 3-99)

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

This command specifies the lower limit for limit tests.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	-1.000000E+00 for most functions; see **Details**

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<Y>`	Limit number: `1` or `2`
`<n>`	The value of the lower limit ($-1E+12$ to $1E+12$)

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

When the diode function is selected, the default is 0.3 for limit 1 and -1 for limit 2.

Example

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

Also see

[:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA]](#) (on page 6-27)

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

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

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

Usage

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

<function>	The function to which the setting applies; see **Functions**
<Y>	Limit number: 1 or 2
<state>	Disable the limit test: OFF or 0
	Enable the limit test: ON or 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

Example

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

Also see

:CALCulate2:<function>:LIMit<Y>:CLEar:AUTO (on page 6-21)
:CALCulate2:<function>:LIMit<Y>:CLEar[:IMMediate] (on page 6-22)
:CALCulate2:<function>:LIMit<Y>:FAIL? (on page 6-23)
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] (on page 6-25)
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] (on page 6-27)

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

This command specifies the upper limit for a limit test.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1.000000E+00 for most functions; see **Details**

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<Y>`	Limit number: `1` or `2`
`<n>`	The value of the upper limit (-1e+12 to 1e+12)

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATIO`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

Default is 0.8 for limit 1 when the diode function is selected; 10 when the continuity function is selected. The default for limit 2 for the diode and continuity functions is 1.

Example

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

Also see

[:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA]](#) (on page 6-25)
[:CALCulate2:<function>:LIMit<Y>:STATe](#) (on page 6-26)

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

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

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

Usage

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

<function>	The function to which the setting applies; see **Functions**
<operation>	The name of the math operation: • y = mx+b: MXB • Percent: PERCent • Reciprocal: RECiprocal

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

You can choose one of the following math operations:

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

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

Example

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

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:MBFactor (on page 6-29)
:CALCulate[1]:<function>:MATH:MMFactor (on page 6-30)
:CALCulate[1]:<function>:MATH:PERCent (on page 6-32)
:CALCulate[1]:<function>:MATH:STATe (on page 6-33)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The offset for the y = mx + b operation; the valid range is −1e12 to +1e12

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

$$y = mx + b$$

Where:

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

Example

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

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:FORMat (on page 6-28)
:CALCulate[1]:<function>:MATH:MMFactor (on page 6-30)
:CALCulate[1]:<function>:MATH:STATe (on page 6-33)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The scale factor; the valid range is −1e12 to +1e12

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

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

$$y = mx + b$$

Where:

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

Example

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

Also see

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The reference used when the math operation is set to percent; the range is -1e12 to +1e12

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

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

Where:

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

Example

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

Also see

Calculations that you can apply to measurements (on page 3-6)
:CALCulate[1]:<function>:MATH:FORMat (on page 6-28)
:CALCulate[1]:<function>:MATH:STATe (on page 6-33)

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

This command enables or disables math operation.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	Disable math operations: `OFF` or `0` Enable math operations: `ON` or `1`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

Example

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

Also see

:CALCulate[1]:<function>:MATH:FORMat (on page 6-28)
Calculations that you can apply to measurements (on page 3-6)

DIGital subsystem

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

:DIGital:LINE<n>:MODE

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

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

Usage

```
:DIGital:LINE<n>:MODE <triggerType>, <lineMode>
:DIGital:LINE<n>:MODE?
```

`<n>`	The digital I/O line (1 to 6)
`<triggerType>`	Sets the digital line control type; the options are: • Allow direct digital control of the line: `DIGital` • Configure for trigger control: `TRIGger` • Configure as a synchronous master or acceptor: `SYNChronous`
`<lineMode>`	The line mode; see **Details** for values

Details

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 acceptor

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

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

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

Example

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

Also see

Digital I/O port configuration (on page 3-45)
:TRIGger:DIGital<n>:IN:EDGE (on page 6-194)

:DIGital:LINE<n>:STATe

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

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

Usage

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

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

Details

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

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

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

Example

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

Also see

Digital I/O port configuration (on page 3-45)
:DIGital:LINE<n>:MODE (on page 6-34)
:DIGital:READ? (on page 6-37)
:DIGital:WRITe <n> (on page 6-38)
:TRIGger:DIGital<n>:IN:EDGE (on page 6-194)

:DIGital:READ?

This command reads the digital I/O port.

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

Usage

```
:DIGital:READ?
```

Details

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

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

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

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

Example

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

Also see

Digital I/O bit weighting (on page 3-53)
Digital I/O port configuration (on page 3-45)

:DIGital:WRITe <n>

This command writes to all digital I/O lines.

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

Usage

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

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

Details

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

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

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

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

Example

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

Also see

Digital I/O bit weighting (on page 3-53)
Digital I/O port configuration (on page 3-45)

DISPlay subsystem

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

:DISPlay:CLEar

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

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

Usage

```
:DISPlay:CLEar
```

Example

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

Also see

:DISPlay:USER<n>:TEXT[:DATA] (on page 6-43)

:DISPlay:<function>:DIGits

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	3.5 digit resolution: `3` 4.5 digit resolution: `4` 5.5 digit resolution: `5` 6.5 digit resolution: `6` 7.5 digit resolution: `7` (not available for digitize functions)

Functions and defaults

Function	Def	Function	Def	Function	Def	Function	Def
VOLTage[:DC]	7	TEMPerature	5	RESistance	7	VOLTage[:DC]:RATIO	7
VOLTage:AC	6	CONTinuity	4	FRESistance	7	DIGitize:VOLTage	4
CURRent[:DC]	7	FREQuency[:VOLTage]	6	DIODe	7	DIGitize:CURRent	4
CURRent:AC	6	PERiod[:VOLTage]	6	CAPacitance	4		

Details

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

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

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

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

NOTE

The digits for the temperature, continuity, frequency, period, and capacitance functions are always set to the default values and cannot be changed.

Example

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

Also see

:FORMat:ASCii:PRECision (on page 6-44)

:DISPlay:LIGHt:STATe

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

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

Usage

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

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

Details

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

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

NOTE

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

Example

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

Also see

[Adjust the backlight brightness and dimmer](#) (on page 2-10)

:DISPlay:READing:FORMat

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

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

Usage

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

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

Details

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

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

Example

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

Also see

[Setting the display format](#) (on page 2-53)

:DISPlay:SCReen

This command changes which front-panel screen is displayed.

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

Usage

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

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

Example

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

Also see

None

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

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

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

Usage

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

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

Details

This command defines text messages for the USER swipe screen.

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

Example

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

Also see

:DISPlay:SCReen (on page 6-42)

FORMat subsystem

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

:FORMat:ASCii:PRECision

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

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

Usage

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

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

Details

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

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

Example

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

Also see

:FORMat[:DATA] (on page 6-46)

:FORMat:BORDer

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

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

Usage

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

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

Details

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

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

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

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

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

(Single precision)

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

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

(Single precision)

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

Example

FORM:BORD NORM	Use the normal byte order.

Also see

:FORMat[:DATA] (on page 6-46)

:FORMat[:DATA]

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

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

Usage

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

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

Details

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

> **NOTE**
>
> The Model DMM7510 only responds to input commands using the ASCII format, regardless of the data format that is selected for output strings.

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

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

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

Example

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

Also see

[:TRACe:DATA?](#) (on page 6-147)

ROUTe subsystem

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

:ROUTe:TERMinals

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

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

Usage

```
:ROUTe:TERMinals?
```

Details

You must use the front-panel TERMINALS button to change which set of terminals the instrument reads.

This query returns the set of input and output terminals that the instrument is using. If the instrument is using the front-panel terminals, the return is:

```
FRON
```

If the instrument is using the rear-panel terminals, the return is:

```
REAR
```

Example

`:ROUT:TERM?`	Query to verify which terminals are used. Output if the rear terminals are used: `REAR`

Also see

None

SCRipt subsystem

The SCRipt subsystem controls macro scripts. For additional information on macro scripts, refer to Saving front-panel settings into a macro script (on page 3-31).

SCRipt:RUN

This command runs a script.

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

Usage

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

`<scriptName>`	The name of the script

Details

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

Example

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

Also see

Saving front-panel settings into a macro script (on page 3-31)
Scripts menu (on page 2-45)

SENSe1 subsystem

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

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

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

This command determines the aperture setting for the selected function.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The time of the aperture; see **Details**

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Function	Default value	Range
Voltage (AC and DC)	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Current (AC and DC)	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Resistance (2-wire and 4-wire)	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Diode	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Temperature	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Frequency and Period	10 ms	10 ms to 0.273 s

Function	Default value	Range
Voltage ratio	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 µs to 0.25 s 10 µs to 0.24 s
Digitize (voltage and current)	Automatic	1 µs to 1 ms set in 1 µs increments

The functionality of aperture depends on whether you are using a measure function or a digitize function.

Aperture for a measure function

If you are using a measure function, the aperture sets the amount of time the ADC takes when making a measurement, which is the integration period for the selected measurement function. The integration period is specified in seconds. In general, a short integration period provides a fast reading rate, while a long integration period provides better accuracy. The selected integration period is a compromise between speed and accuracy.

During the integration period, if an external trigger is sent, the trigger is ignored.

You can also set the integration rate by setting the number of power line cycles (NPLCs). Changing the NPLC value changes the aperture time and changing the aperture time changes the NPLC value.

To calculate the aperture based on the NPLC value, use the following formula.

$$\text{Aperture} = \frac{\text{NPLC}}{f}$$

where:

- Aperture is the integration rate in seconds for each integration
- NPLC is the number of power line cycles for each integration
- f is the power line frequency

If you set the NPLCs, the aperture setting changes to reflect that value. If you set the aperture, the NPLC setting is changed.

For the AC voltage and AC current functions, if the detector bandwidth setting is set to 3 Hz or 30 Hz, the aperture value is fixed and cannot be changed.

NOTE

If line synchronization is enabled, the integration period does not start until the beginning of the next power line cycle. For example, if a reading is triggered at the positive peak of a power line cycle, the integration period does not start until that power line cycle is completed. The integration period starts when the positive-going sine wave crosses zero volts.

To see the line frequency that is auto-detected by the instrument, use the `:SYSTem:LFRequency?` command.

Aperture for a digitize function

If you are using a digitize function, the aperture determines how long the instrument makes measurements. The aperture is set to automatic or to a specific value in 1 μs intervals.

The aperture is the actual acquisition time of the instrument on the signal. It must be less than the set sample rate. The minimum aperture is 1 μs when the maximum sampling rate is 1,000,000 samples per second.

When the aperture is set to automatic, the aperture is equivalent to the sample rate interval.

If you specify an aperture and the value is less than 1 μs, it is rounded down to the nearest micro second resolution.

If you set a value that is longer than the sample rate interval, the instrument generates an error event.

Set the sample rate before changing the aperture.

The maximum aperture available is 1 divided by the sample rate. The aperture cannot be set to more than this value.

When automatic is selected, the aperture setting is set to the maximum value possible for the selected sample rate. You select automatic by sending AUTO or 0.

Example

| ```
DIG:FUNC "CURR"
DIG:CURR:SRATE 1000000
DIG:CURR:APER AUTO
DIG:COUN 10
MEAS:DIG?
``` | Set the digitize function to measure current. Set the sample rate to 1,000,000, with a count of 10, and automatic aperture. Make a digitize measurement. |

Also see

[:SENSe[1]]:<function>:NPLCycles (on page 6-81)
[:SENSe[1]]:<function>:SRATe (on page 6-99)
:SYSTem:LFRequency? (on page 6-141)

[:SENSe[1]]:<function>:ATRigger:EDGE:LEVel

This command defines the signal level that generates the trigger event for the edge trigger mode.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:EDGE:LEVel <setting>
[:SENSe[1]]:<function>:ATRigger:EDGE:LEVel?
```

<function>	The function to which the setting applies; see **Functions**
<setting>	The signal level that generates the trigger event

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command is only available when the analog trigger mode is set to edge.

The edge level can be set to any value in the active measurement range. See the Model DMM7510 specifications for more information on the resolution and accuracy of the analog trigger.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

`FUNC "CURR"`	Set measure function to DC current.
`CURR:RANGE 3`	Set range to 3 A.
`CURR:AZER OFF`	Disable autozero.
`CURR:ATR:MODE EDGE`	Set the analog trigger mode to edge.
`CURR:ATR:EDGE:LEV 2.5`	Set the analog trigger level to 2.5 A.

Also see

[:SENSe[1]]:<function>:ATRigger:EDGE:SLOPe

This command defines the slope of the analog trigger edge.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:EDGE:SLOPe <setting>
[:SENSe[1]]:<function>:ATRigger:EDGE:SLOPe?
```

<function>	The function to which the setting applies; see **Functions**
<setting>	The direction: • Rising: RISing • Falling: FALLing

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This is only available when the analog trigger mode is set to edge.

Rising causes a trigger event when the analog signal trends from below the analog signal level to above the level.

Falling causes a trigger event when the signal trends from above to below the level.

Example

FUNC "CURR" CURR:RANGE 3 CURR:AZER OFF CURR:ATR:MODE EDGE CURR:ATR:EDGE:LEV 2.5 CURR:ATR:EDGE:SLOP RIS	Set measure function to DC current. Set range to 3 A. Disable autozero. Set the analog trigger mode to edge. Set the analog trigger level to 2.5 A. Set the analog trigger slope to rising.

Also see

[:SENSe[1]]:<function>:ATRigger:HFReject

This command enables or disables high frequency rejection on analog trigger events.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:HFReject <setting>
[:SENSe[1]]:<function>:ATRigger:HFReject?
```

<function>	The function to which the setting applies; see **Functions**
<setting>	0 µs: OFF or 0 60 µs: ON or 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

False triggering around the set analog trigger level may occur with low frequency signals that are noisy, DC, or have low amplitude and slew rate during the peaks of input sine waves less than 250 Hz. High frequency rejection avoids the false triggers by the requiring the trigger event to be sustained for at least 60 µs. This behavior is similar to a low pass filter effect with a 4 kHz 3 dB bandwidth.

When high frequency rejection is on, 60 µs of additional trigger latency is incurred. You may also need to adjust the trigger levels to ensure that trigger condition is satisfied for at least 60 µs.

Example

VOLT:ATR:HFR ON	Turn high frequency rejection for the analog trigger to on when the measure function is set to DC voltage.

Also see

[Analog triggering overview](#) (on page 3-60)

[:SENSe[1]]:<function>:ATRigger:MODE

This command configures the type of signal behavior that can generate a trigger event.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:MODE <setting>
[:SENSe[1]]:<function>:ATRigger:MODE?
```

<function>	The function to which the setting applies; see **Functions**
<setting>	The setting: • Edge (signal crosses one level): `EDGE` • Pulse (two complementary edge events meet a specified time constraint): `PULse` • Window (signal enters or exits a window defined by two levels): `WINDow` • No analog triggering: `OFF`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

When edge is selected, the trigger occurs when the signal crosses a certain level. You also specify if the trigger occurs on the rising or falling edge of the signal.

When pulse is selected, the trigger occurs when a pulse passes through the specified level and meets the constraint that you set on its width. You also specify the polarity of the signal (above or below the trigger level).

When window is selected, the trigger occurs when the signal enters or exits the window defined by the low and high signal levels.

Example

FUNC "CURR"	Set measure function to DC current.
CURR:RANGE 3	Set range to 3 A.
CURR:AZER OFF	Disable autozero.
CURR:ATR:MODE EDGE	Set the analog trigger mode to edge.
CURR:ATR:EDGE:LEV 2.5	Set the analog trigger level to 2.5 A.
CURR:ATR:EDGE:SLOP RIS	Set the analog trigger slope to rising.

Also see

Analog triggering overview (on page 3-60)

[:SENSe[1]]:<function>:ATRigger:PULSe:CONDition

This command defines if the pulse must be greater than or less than the pulse width before a trigger is generated.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:PULSe:CONDition <setting>
[:SENSe[1]]:<function>:ATRigger:PULSe:CONDition?
```

`<function>`	The function to which the setting applies; see **Functions**
`<setting>`	The setting: • The pulse width must be greater than the specified pulse width: `GReater` • The pulse width must be less than the specified pulse width: `LESS`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Only available when the analog trigger mode is set to pulse.

Example

CURR:ATR:MODE PULSe CURR:ATR:PULSe:COND LESS	Set the analog trigger for the current function to pulse. Set the condition to less.

Also see

[:SENSe[1]]:<function>:ATRigger:MODE (on page 6-55)
[:SENSe[1]]:<function>:ATRigger:PULSe:WIDTh (on page 6-59)
Analog triggering overview (on page 3-60)

[:SENSe[1]]:<function>:ATRigger:PULSe:LEVel

This command defines the pulse level that generates a trigger event.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:PULSe:LEVel <value>
[:SENSe[1]]:<function>:ATRigger:PULSe:LEVel?
```

<function>	The function to which the setting applies; see **Functions**
<value>	The signal level

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Only available when the analog trigger mode is set to pulse.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

FUNC "CURR" CURR:RANGE 3 CURR:AZER OFF CURR:ATR:MODE PULSE CURR:ATR:PULS:LEV 2.5	Set measure function to DC current. Set range to 3 A. Disable autozero. Set the analog trigger mode to pulse. Set the analog trigger level to 2.5 A.

Also see

[:SENSe[1]]:<function>:ATRigger:MODE (on page 6-55)
Analog triggering overview (on page 3-60)

[:SENSe[1]]:<function>:ATRigger:PULSe:POLarity

This command defines the polarity of the pulse that generates an analog trigger event.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	ABOVe

Usage

```
[:SENSe[1]]:<function>:ATRigger:PULSe:POLarity <setting>
[:SENSe[1]]:<function>:ATRigger:PULSe:POLarity?
```

`<function>`	The function to which the setting applies; see **Functions**
`<setting>`	The setting: • Above: `ABOVe` • Below: `BELow`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Only used when analog trigger mode is pulse.

Determines if the trigger occurs when the pulse is above the defined signal level or below the defined signal level.

Example

`FUNC "CURR"`	Set measure function to DC current.
`CURR:RANGE 3`	Set range to 3 A.
`CURR:AZER OFF`	Disable autozero.
`CURR:ATR:MODE PULSE`	Set the analog trigger mode to pulse.
`CURR:ATR:PULS:LEV 2.5`	Set the analog trigger level to 2.5 A.
`CURR:ATR:PULS:POL BEL`	Set the polarity to below.

Also see

[:SENSe[1]]:<function>:ATRigger:PULSe:WIDTh

This command defines the threshold value for the pulse width.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	20 μs

Usage

```
[:SENSe[1]]:<function>:ATRigger:PULSe:WIDTh <setting>
[:SENSe[1]]:<function>:ATRigger:PULSe:WIDTh?
```

<function>	The function to which the setting applies; see **Functions**
<setting>	The threshold value for the pulse width: 1 μs to 40 ms

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This option is only available when the analog trigger mode is set to pulse.

This option sets either the minimum or maximum pulse width that generates a trigger event. The value of pulse condition determines whether this value is interpreted as the minimum or maximum pulse width.

Example

FUNC "CURR" CURR:RANGE 3 CURR:AZER OFF CURR:ATR:MODE PULSE CURR:ATR:PULS:LEV 2.5 CURR:ATR:PULS:POL BEL CURR:ATR:PULS:WIDT 30e-6	Set measure function to DC current. Set range to 3 A. Disable autozero. Set the analog trigger mode to pulse. Set the analog trigger level to 2.5 A. Set the polarity to below. Set the analog trigger pulse width to 30 μs.

Also see

Analog triggering overview (on page 3-60)
[:SENSe[1]]:<function>:ATRigger:MODE (on page 6-55)
[:SENSe[1]]:<function>:ATRigger:PULSe:CONDition (on page 6-56)
[:SENSe[1]]:<function>:ATRigger:PULSe:LEVel (on page 6-57)

[:SENSe[1]]:<function>:ATRigger:WINDow:DIRection

This command defines if the analog trigger occurs when the signal enters or leaves the defined upper and lower analog signal level boundaries.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:WINDow:DIRection <setting>
[:SENSe[1]]:<function>:ATRigger:WINDow:DIRection?
```

`<function>`	The function to which the setting applies; see **Functions**
`<setting>`	The direction: • Enter: `ENTer` • Leave: `LEAVe`

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATIO`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

This is only available when the analog trigger mode is set to window.

Example

`FUNC "CURR"`
`CURR:RANGE 3`
`CURR:AZER OFF`
`CURR:ATR:MODE WINDOW`
`CURR:ATR:WIND:LEV:HIGH 2.5`
`CURR:ATR:WIND:LEV:LOW 1`
`CURR:ATR:WIND:DIR LEAV`
	Set measure function to DC current.
	Set range to 3 A.
	Disable autozero.
	Set the analog trigger mode to window.
	Set the analog trigger level for the low point of the window to 1.0 A and the high point for 2.5 A.
	Set the trigger to occur when the signal leaves the window (signal below 1.0 A or above 2.5 A).

Also see

[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:HIGH

This command defines the upper boundary of the analog trigger window.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	DC current and digitize current: 5e-06 DC voltage and digitize voltage: 0.05

Usage

```
[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:HIGH <value>
[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:HIGH?
```

<function>	The function to which the setting applies; see **Functions**
<value>	The upper boundary of the window

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Only available when the analog trigger mode is set to window.

The high level must be greater than the low level.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

| ```
FUNC "CURR"
CURR:RANGE 3
CURR:AZER OFF
CURR:ATR:MODE WINDOW
CURR:ATR:WIND:LEV:HIGH 2.5
CURR:ATR:WIND:LEV:LOW 1
CURR:ATR:WIND:DIR LEAV
``` | Set measure function to DC current.<br>Set range to 3 A.<br>Disable autozero.<br>Set the analog trigger mode to window.<br>Set the analog trigger level for the low point of the window to 1.0 A and the high point for 2.5 A.<br>Set the trigger to occur when the signal leaves the window (signal below 1.0 A or above 2.5 A). |

Also see

Analog triggering overview (on page 3-60)
[:SENSe[1]]:<function>:ATRigger:MODE (on page 6-55)
[:SENSe[1]]:<function>:ATRigger:WINDow:DIRection (on page 6-60)
[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:LOW (on page 6-62)

[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:LOW

This command defines the lower boundary of the analog trigger window.

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

Usage

```
[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:LOW <value>
[:SENSe[1]]:<function>:ATRigger:WINDow:LEVel:LOW?
```

<function>	The function to which the setting applies; see **Functions**
<value>	The lower boundary of the window

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Only available when the analog trigger mode is set to window.

The low level must be less than the high level.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

`FUNC "CURR"`
`CURR:RANGE 3`
`CURR:AZER OFF`
`CURR:ATR:MODE WINDOW`
`CURR:ATR:WIND:LEV:HIGH 2.5`
`CURR:ATR:WIND:LEV:LOW 1`
`CURR:ATR:WIND:DIR LEAV`
	Set measure function to DC current.
	Set range to 3 A.
	Disable autozero.
	Set the analog trigger mode to window.
	Set the analog trigger level for the low point of the window to 1.0 A and the high point for 2.5 A.
	Set the trigger to occur when the signal leaves the window (signal below 1.0 A or above 2.5 A).

Also see

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The number of readings required for each filtered measurement (1 to 100)

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

Example 1

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

Example 2

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

Also see

Filtering measurement data (on page 3-10)
[:SENSe[1]]:<function>:AVERage[:STATe] (on page 6-64)
[:SENSe[1]]:<function>:AVERage:TCONtrol (on page 6-65)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<state>`	The filter status; set to one of the following values: • Disable the averaging filter: `OFF` or `0` • Enable the averaging filter: `ON` or `1`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

Example 1

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

Example 2

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

Example 3

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

Also see

Filtering measurement data (on page 3-10)
[:SENSe[1]]:<function>:AVERage:COUNt (on page 6-63)
[:SENSe[1]]:<function>:AVERage:TCONtrol (on page 6-65)
[:SENSe[1]]:<function>:AVERage:WINDow (on page 6-67)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<type>`	The filter type to use when filtering is enabled; set to one of the following values: • Repeating filter: `REPeat` • Moving filter: `MOVing`

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATIO`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

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

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

NOTE

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

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

Example 1

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

Example 2

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

Example 3

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

Also see

Filtering measurement data (on page 3-10)
[:SENSe[1]]:<function>:AVERage:COUNt (on page 6-63)
[:SENSe[1]]:<function>:AVERage[:STATe] (on page 6-64)

[:SENSe[1]]:\<function\>:AVERage:WINDow

This command sets the window for the averaging filter that is used for measurements for the selected function.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0.1

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The filter window setting; the range is between 0 and 10 to indicate percent of range

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATIO`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

This command selects the window size for the averaging filter.

The noise window allows a faster response time to large signal step changes. A reading that falls outside the plus or minus noise window fills the filter stack immediately.

If the noise does not exceed the selected percentage of range, the reading is based on an average of reading conversions — the normal averaging filter. If the noise does exceed the selected percentage, the reading is a single reading conversion, and new averaging starts from this point.

Example 1

`CURR:AVER:COUNT 10`
`CURR:AVER:TCON MOV`
`CURR:AVER:WIND 5`
`CURR:AVER ON`
	Set the averaging filter type to moving average, with a filter count of 10 with a window of 5 %. Enable the averaging filter.

Also see

Filtering measurement data (on page 3-10)
[:SENSe[1]]:\<function\>:AVERage:COUNt (on page 6-63)
[:SENSe[1]]:\<function\>:AVERage[:STATe] (on page 6-64)
[:SENSe[1]]:\<function\>:AVERage:TCONtrol (on page 6-65)

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

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

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

Usage

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

<function>	The function to which the setting applies; see **Functions**
<state>	The status of autozero: • Disable autozero: OFF or 0 • Enable autozero: ON or 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

To ensure the accuracy of readings, the instrument must periodically get new measurements of its internal ground and voltage reference. The time interval between updates to these reference measurements is determined by the integration aperture that is being used for measurements. The Model DMM7510 uses separate reference and zero measurements for each aperture.

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

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

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

For AC voltage and AC current measurements where the detector bandwidth is set to 3 Hz or 30 Hz, autozero is set on and cannot be changed.

Example

VOLT:AZER OFF	Sets autozero off for voltage measurements.

Also see

Automatic reference measurements (on page 2-147)
[:SENSe[1]]:AZERo:ONCE (on page 6-110)

[:SENSe[1]]:<function>:BIAS:ACTual?

This command returns the amount of current the instrument is sourcing when it makes measurements.

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

Usage

```
[:SENSe[1]]:<function>:BIAS:ACTual?
```

<function>	The function to which the setting applies; see **Functions**

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Returns the bias level: 10 μA, 100 μA, or 1 mA.

Reads the actual amount of current that is sourced by the instrument when a measurement is made.

Example

:FUNC "DIOD" :DIOD:BIAS:ACTUAL?	Set the function to diode. Query the actual bias level that is used; example output: 0.00100351

Also see

[:SENSe[1]]:<function>:BIAS:LEVel (on page 6-70)

[:SENSe[1]]:<function>:BIAS:LEVel

This command selects the amount of current the instrument sources when it makes measurements.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The bias level: 10 µA, 100 µA, or 1 mA

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Selects the amount of current that is sourced by the instrument to make measurements.

Example

`DIOD:BIAS:LEVel 0.0001`	For the diode functions, sets a bias level of 100 µA.

Also see

[:SENSe[1]]:<function>:BIAS:ACTual? (on page 6-69)

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

This command determines if AC or DC signal coupling is used.

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

Usage

```
[:SENSe[1]]:<function>:COUPling <type>
[:SENSe[1]]:<function>:COUPling?
```

`<function>`	The function to which the setting applies; see **Functions**
`<type>`	The type of coupling: • AC • DC

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command selects the type of input coupling that is used for the selected function.

When DC is selected, the instrument measures AC and DC components of the signal. When AC is selected, the instrument only measures the AC components of the signal.

If AC coupling is selected, you can change input impedance settings, but they do not take effect until DC coupling is selected.

Example

`DIG:VOLT:COUP AC`	Set the digitize volt coupling to AC.

Also see

DC and AC coupling (on page 2-129)
[:SENSe[1]]:<function>:COUPling:AC:FILTer (on page 6-72)
[:SENSe[1]]:<function>:COUPling:AC:FREQuency (on page 6-73)
[:SENSe[1]]:<function>:INPutimpedance (on page 6-79)

[:SENSe[1]]:<function>:COUPling:AC:FILTer

This command selects the instrument settling time when coupling is set to AC.

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

Usage

```
[:SENSe[1]]:<function>:COUPling:AC:FILTer <type>
[:SENSe[1]]:<function>:COUPling:AC:FILTer?
```

`<function>`	The function to which the setting applies; see **Functions**
`<type>`	Type of AC coupling filter: • Slow (800 ms): `SLOW` • Fast (80 ms): `FAST`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This option is only used when digitize signal coupling is set to AC.

When the signal coupling is set to AC, there may still be some DC signal content that comes in with the AC signal. To allow this signal to settle out, you can set AC coupling filter to slow. When the filter is set to slow, the instrument adds an 800 ms delay before making measurements.

When the AC coupling filter is set to fast, the instrument adds an 80 ms delay before making measurements. Set the AC filter to fast for faster settling when measuring rapidly changing inputs. For most digitize voltage measurements, the 80 ms delay is enough time for the range to settle.

Example

`DIG:VOLT:COUP AC` `DIG:VOLT:COUP:AC:FILT FAST`	Set the coupling type to AC. Set the filter to fast.

Also see

DC and AC coupling (on page 2-129)
[:SENSe[1]]:<function>:COUPling (on page 6-71)

[:SENSe[1]]:<function>:COUPIing:AC:FREQuency

This command allows you to optimize the amplitude to compensate for signal loss across the coupling capacitor when AC coupling is selected.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1000

Usage

```
[:SENSe[1]]:<function>:AC:FREQuency <range>
[:SENSe[1]]:<function>:AC:FREQuency?
```

`<function>`	The function to which the setting applies; see **Functions**
`<range>`	The frequency: 3 Hz to 1 MHz

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command is only used when the digitize coupling type is set to AC.

For example, if you are measuring a 50 Hz signal, you could set this to 50 Hz to compensate for voltage drop across the coupling capacitor.

Example

DIG:VOLT:COUP AC DIG:VOLT:COUP:AC:FILT FAST DIG:VOLT:COUP:AC:FREQ 500	Set the coupling type to AC. Set the filter to fast. Set the frequency to 500 Hz.

Also see

[:SENSe[1]]:<function>:COUPIing (on page 6-71)

[:SENSe[1]]:<function>:DB:REFerence

This command defines the decibel (dB) reference setting for the DMM in volts.

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

Usage

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

<function>	The function to which the setting applies; see **Functions**
<n>	The decibel reference range: • DC voltage and digitize voltage: 1e-7 V to 1000 V • AC voltage: 1e-7 V to 700 V

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This value only applies when the unit setting for the function is set to decibels.

Example

FUNC "VOLT" VOLT:UNIT DB VOLT:DB:REF 5	Sets the units to decibel and sets the dB reference to 5 for DC volts.

Also see

[:SENSe[1]]:<function>:UNIT (on page 6-109)

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

This command enables or disables the dry circuit feature of the 4-wire resistance measure function.

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

Usage

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

`<function>`	See **Functions**
`<state>`	Disable: `0` or `OFF`; available for all ranges
	Enable: `1` or `ON`; available for 1 Ω to 10 kΩ ranges

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	**FRESistance**	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Enabling dry circuit limits the open-circuit voltage to below 20 mV, which is often required with low-glitch measurements, such as measuring switch and relay contact resistance.

When dry circuit is enabled, offset compensation is automatically enabled.

Example

`FUNC "FRES"` `FRES:DCIR ON`	Set the measure function to 4-wire resistance and enable dry circuit.

Also see

[:SENSe[1]]:<function>:OCOMpensated (on page 6-83)
[:SENSe[1]]:<function>:RANGe[:UPPer] (on page 6-86)

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

This command enables or disables the automatic delay that occurs before each measurement.

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

Usage

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

<function>	The function to which the setting applies; see **Functions**
<state>	Disable the auto delay: OFF Enable the auto delay: ON

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

When this is enabled, a delay is added before each measurement.

Example

CURR:DEL:AUTO OFF	Turn off auto delay when DC current is measured.

Also see

[:SENSe[1]]:<function>:DELay:USER<n> (on page 6-77)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The user delay to which this time applies (1 to 5)
`<delayTime>`	The delay (0 for no delay, or 167 ns to 10,000 s)

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

To use this command in a trigger model, assign the delay to the dynamic delay block.
The delay is specific to the selected function.

Example

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

Also see

:TRIGger:BLOCk:DELay:DYNamic (on page 6-184)

[:SENSe[1]]:<function>:DETector:BANDwidth

This command selects the detector bandwidth for AC current and AC voltage measurements.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	30

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	3 Hz, 30 Hz, or 300 Hz

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

You can set the detector bandwidth to improve measurement accuracy. Select the bandwidth that contains the lowest frequency component of the input signal. For example, if the lowest frequency component of your input signal is 40 Hz, use a bandwidth setting of 30 Hz.

If the bandwidth is set to 3 Hz or 30 Hz, the autozero feature is always enabled and the integration rate is fixed.

Example

`FUNC "VOLT:AC"` `VOLT:AC:DET:BAND 30`	Set the measure function to AC volts. Set the detector bandwidth for AC volts to 30 Hz.

Also see

[:SENSe[1]]:<function>:APERture (on page 6-49)
[:SENSe[1]]:<function>:AZERo[:STATe] (on page 6-68)
[:SENSe[1]]:<function>:NPLCycles (on page 6-81)
[:SENSe[1]]:AZERo:ONCE (on page 6-110)

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

This command determines when the 10 MΩ input divider is enabled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	MOHM10

Usage

```
[:SENSe[1]]:<function>:INPutimpedance <n>
[:SENSe[1]]:<function>:INPutimpedance?
```

<function>	The function to which the setting applies; see **Functions**
<n>	10 MΩ for all ranges: MOHM10 Automatic: AUTO

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Automatic input impedance provides the lowest measure noise with the highest isolation on the device under test (DUT). When automatic input impedance is selected, the 100 mV to 10 V voltage ranges have more than 10 GΩ input impedance. For the 100 V and 1000 V ranges, a 10 MΩ input divider is placed across the HI and LO input terminals.

When the input impedance is set to 10 MΩ, the 100 mV to 1000 V ranges have a 10 MΩ input divider across the HI and LO input terminals. The 10 M impedance provides stable measurements when the terminals are open (approximately 100 μV at 1 PLC).

Choosing automatic input impedance is a balance between achieving low DC voltage noise on the 100 mV and 1 V ranges and optimizing measurement noise due to charge injection. The Model DMM7510 is optimized for low noise and charge injection when the DUT has less than 100 KΩ input resistance. When the DUT input impedance is more than 100 K, selecting an input impedance of 10 MΩ optimizes the measurement for lowest noise on the 100 mV and 1 V ranges. You can achieve short-term low noise and low charge injection on the 100 mV and 1 V ranges with autozero off. For the 10 V to 1000 V ranges, both input impedance settings achieve low charge injection.

For the digitize voltage function, the input impedance setting is only available when coupling is set to DC.

Example

:DIG:VOLT:INP AUTO	Set input impedance to be set automatically when the digitize voltage function is selected.

Also see

[:SENSe[1]]:<function>:COUPling (on page 6-71)

[:SENSe[1]]:<function>:LINE:SYNC

This command determines if line synchronization is used during the measurement.

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

Usage

```
[:SENSe[1]]:<function>:LINE:SYNC <state>
[:SENSe[1]]:<function>:LINE:SYNC?
```

`<function>`	The function to which the setting applies; see **Functions**
`<state>`	Disable: `OFF` or `0` Enable: `ON` or `1`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

When line synchronization is enabled, measurements are initiated at the first positive-going zero crossing of the power line cycle after the trigger.

Example

`CURR:LINE:SYNC ON`	Turn on line synchronization when DC current is measured.

Also see

Line cycle synchronization (on page 4-1)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The number of power-line cycles for each measurement: 0.0005 to 15 (60 Hz) or 12 (50 Hz)

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

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

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

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

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

NOTE

The measurement time can also be set as an aperture time. Changing the NPLC value changes the aperture time and changing the aperture time changes the NPLC value.

Example 1

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

Example 2

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

Example 3

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

Also see

[:SENSe[1]]:<function>:APERture (on page 6-49)
Using aperture or NPLCs to adjust speed and accuracy (on page 4-2)

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

This command enables or disables offset compensation.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	Temperature, 3- or 4-wire RTD: ON (1) 4-wire resistance: OFF (0)

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<state>`	Disable offset compensation: `OFF` or `0` Enable offset compensation: `ON` or `1` (for 4-wire resistance, not available for ranges more than 1 MΩ)

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

For 4-wire resistance measurements, when offset compensation is enabled, the measure range is limited to a maximum of 100 kΩ. Offset compensation is automatically enabled when dry circuit is enabled.

For 2-wire resistance measurements, offset compensation is always set to off.

For temperature measurements, offset compensation is only available when the transducer type is set to 3-wire or 4-wire RTD.

Example

``*RST
:SENS:FUNC "FRES"
:SENS:FRES:RANG 10e3
:FRES:OCOM ON
:COUNT 5
:TRAC:TRIG "defbuffer1"
:TRAC:DATA? 1, 5, "defbuffer1", READ```	Reset the instrument. Set the measurement function to 4-wire resistance and set the range to 10 kΩ. Turn offset-compensated ohms on. Set the measurement count to 5. Make measurements and store them in `defbuffer1`. Retrieve the measurement values for readings 1 to 5.

Also see

Offset-compensated ohms (on page 2-108)

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

This command determines if the detection of open leads is enabled or disabled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	4W Res: OFF (0) Temperature: ON (1)

Usage

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

<function>	The function to which the setting applies; see **Functions**
<state>	Disable: OFF or 0 Enable: ON or 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

For temperature measurements, this is only available when the transducer is set to a thermocouple or one of the RTDs.

Long lengths of thermocouple wire can have a large amount of capacitance, which is seen at the input of the DMM. If an intermittent open occurs in the thermocouple circuit, the capacitance can cause an erroneous on-scale reading. The open thermocouple detection circuit, when enabled, applies a 100 μA pulse of current to the thermocouple before the start of each temperature measurement.

Example

TEMP:TRAN TC TEMP:TC:TYPE K TEMP:UNIT C TEMP:ODET OFF	Set the transducer type to thermocouple. Set the thermocouple type to K. Set the units to Celsius. Turn open lead detection off.

Also see

None

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<state>`	Set the measurement range manually: `OFF` or `0`
	Set the measurement range automatically: `ON` or `1`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Auto range selects the best range in which to measure the signal that is applied to the input terminals of the instrument. When auto range is enabled, the range increases at 120 % of range and decreases occurs when the reading is <10 % of nominal range. For example, if you are on the 1 volt range and auto range is enabled, the instrument auto ranges up to the 10 volt range when the measurement exceeds 1.2 volts. It auto ranges down to the 100 mV range when the measurement falls below 1 volt.

This command determines how the range is selected.

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

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

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

NOTE

When the TERMINALS switch is set to REAR and autorange is enabled, autoranging is limited to ranges up to 3 A ranges. The 10 A range is not included in the autorange algorithm.

Example

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

Also see

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

This command determines the positive full-scale measure range.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	Set this command to a specific value or a preset value: See **Details**

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

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

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

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

NOTE
When you set a value for the measurement range, the measurement autorange setting is automatically disabled for the selected measurement function. The range for measure functions defaults to autorange. When you switch from autorange to range, the range is set to the last selected autorange value.

The following table lists the ranges for each function.

If the measurement function is…	The available ranges are…
DC voltage	100 mV, 1 V, 10 V, 100 V, 1000 V
AC voltage	100 mV, 1 V, 10 V, 100 V, 700 V
DC current	10 µA, 100 µA, 1 mA, 10 mA, 100 mA, 1 A, 3 A 10 A available for rear terminals
AC current	1 mA, 10 mA, 100 mA, 1 A, 3 A 10 A available for rear terminals
2-wire resistance	10 Ω, 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ, 1 MΩ, 10 MΩ, 100 MΩ, 1 GΩ
4-wire resistance	1 Ω, 10 Ω, 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ, 1 MΩ, 10 MΩ, 100 MΩ, 1 GΩ
Continuity	1 kΩ (fixed)
Diode	10 V (fixed)
Capacitance	1 nF, 10 nF, 100 nF, 1 µF, 10 µF, 100 µF, 1 mF
DC voltage ratio	100 mV, 1 V, 10 V, 100 V, 1000 V
Digitize voltage	100 mV, 1 V, 10 V, 100 V, 1000 V
Digitize current	10 µA, 100 µA, 1 mA, 10 mA, 100 mA, 1 A, 3 A 10 A available for rear terminals

Example 1

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

Example 2

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

Example 3

`:DIG:VOLT:RANG 50e-3`	Select the 100 mV range.

Also see

Ranges (on page 3-4)
[:SENSe[1]]:<function>:RANGe:AUTO (on page 6-85)

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

This command contains the relative offset value.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The relative offset value; see **Details**

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

The ranges for the relative offset values are listed in the following table.

	Minimum	Maximum
DC voltage	-1000	1000
AC voltage	-700	700
DC current (rear terminals selected)	-10	10
DC current (front terminals selected)	-3	3
AC current (rear terminals selected)	-10	10
AC current (front terminals selected)	-3	3
Resistance	-1e+09	1e+09
4-wire resistance	-1e+09	1e+09
Diode	-10	10
Capacitance	-0.001	0.001
Temperature	-3310	3310
Continuity	-1000	1000
Frequency	-1e+06	1e+06
Period	-1	1
DC voltage ratio - Method set to result	-1E+12	1E+12
DC voltage ratio - Method set to parts	-1000	1000
Digitize voltage	-1000	1000
Digitize current (rear terminals selected)	-10	10
Digitize current (front terminals selected)	-3	3

Example 1

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

Also see

Relative offset (on page 3-5)
[:SENSe[1]]:<function>:RELative:ACQuire (on page 6-90)
[:SENSe[1]]:<function>:RELative:STATe (on page 6-92)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

If an error event occurs during the measurement or overflow, the relative offset level remains at the last valid setting.

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

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

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

Example

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

Also see

[:SENSe[1]]:<function>:RELative (on page 6-88)
[:SENSe[1]]:<function>:RELative:STATe (on page 6-92)

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

This command determines if relative offset is applied to the measurements before calculating the DC voltage ratio value.

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

Usage

```
[:SENSe[1]]:<function>:RELative:METHod <n>
[:SENSe[1]]:<function>:RELative:METHod?
```

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	Apply relative offset: • After calculating the DC voltage ratio value: `RESult` • Before calculating the DC voltage ratio value: `PARTs`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command determines if relative offset is applied to the voltage measurements before the ratio calculation or if the relative offset is applied to the final calculated value.

When the parts methods is selected, the individual readings each have the relative offset value applied before being used to calculate the measurement reading. When parts is selected, the relative offset value is working with smaller ranges, so an error may occur. Reduce the relative offset value if you receive an error. A relative offset is applied to the sense value and then to the input value.

When the results method is selected, the individual readings do not have the relative offset value applied. The relative offset value is applied to the final calculation.

Example

`:FUNC "VOLT:RAT"` `:VOLT:RAT:REL:METH PART`	Set the measure function to DC voltage ratio. Set the method to apply relative offset before generating the ratio.

Also see

Relative offset (on page 3-5)
[:SENSe[1]]:<function>:RELative:ACQuire (on page 6-90)
[:SENSe[1]]:<function>:RELative:STATe (on page 6-92)

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

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

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

Usage

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

<function>	The function to which the setting applies; see **Functions**
<state>	Disable the relative offset: OFF or 0 Enable the relative offset: ON or 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

Displayed reading = Actual measured reading - Relative offset value

Example

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

Also see

Relative offset (on page 3-5)
[:SENSe[1]]:<function>:RELative (on page 6-88)
[:SENSe[1]]:<function>:RELative:ACQuire (on page 6-90)

[:SENSe[1]]:<function>:RTD:ALPHa

This command contains the alpha value of a user-defined RTD.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0.00385055

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	0 to 0.01

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This attribute is only valid when:
- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

`:FUNC "TEMP"`
`:TEMP:TRANsducer TRTD`
`:TEMP:RTD:THR USER`
`:TEMP:RTD:ALPH 0.00385`
`:TEMP:RTD:ZERO 120`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to User.
	Set the alpha RTD value to 0.00385.
	Set the zero RTD value to 120.

Also see

[:SENSe[1]]:<function>:RTD:FOUR (on page 6-96)
[:SENSe[1]]:<function>:RTD:THRee (on page 6-97)
[:SENSe[1]]:<function>:TRANsducer (on page 6-108)

[:SENSe[1]]:<function>:RTD:BETA

This command contains the beta value of a user-defined RTD.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	0.10863

Usage

```
[:SENSe[1]]:<function>:RTD:BETA <value>
[:SENSe[1]]:<function>:RTD:BETA DEFault
[:SENSe[1]]:<function>:RTD:BETA MINimum
[:SENSe[1]]:<function>:RTD:BETA MAXimum
[:SENSe[1]]:<function>:RTD:BETA?
[:SENSe[1]]:<function>:RTD:BETA? DEFault
[:SENSe[1]]:<function>:RTD:BETA? MINimum
[:SENSe[1]]:<function>:RTD:BETA? MAXimum
```

`<function>`	The function to which the setting applies; see **Functions**
`<value>`	0 to 1

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This attribute is only valid when:

- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

`:FUNC "TEMP"`
`:TEMP:TRANsducer TRTD`
`:TEMP:RTD:THR USER`
`:TEMP:RTD:ALPH .005`
`:TEMP:RTD:DELT .00385`
`:TEMP:RTD:ZERO 120`
`:TEMP:RTD:BETA .3`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to User.
	Set the alpha RTD value to 0.005.
	Set the delta RTD value to 0.00385.
	Set the zero RTD value to 120.
	Set the beta RTD value to 0.3.

Also see

[:SENSe[1]]:<function>:RTD:FOUR (on page 6-96)
[:SENSe[1]]:<function>:RTD:THRee (on page 6-97)
[:SENSe[1]]:<function>:TRANsducer (on page 6-108)

[:SENSe[1]]:<function>:RTD:DELTa

This command contains the delta value of a user-defined RTD.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1.4999

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	0 to 5

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This attribute is only valid when:
- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

`:FUNC "TEMP"`
`:TEMP:TRANsducer TRTD`
`:TEMP:RTD:THR USER`
`:TEMP:RTD:ALPH .005`
`:TEMP:RTD:DELT .00385`
`:TEMP:RTD:ZERO 120`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to User.
	Set the alpha RTD value to 0.005.
	Set the delta RTD value to 0.00385.
	Set the zero RTD value to 120.

Also see

[:SENSe[1]]:<function>:RTD:FOUR (on page 6-96)
[:SENSe[1]]:<function>:RTD:THRee (on page 6-97)
[:SENSe[1]]:<function>:TRANsducer (on page 6-108)

[:SENSe[1]]:\<function>:RTD:FOUR

This command contains the type of 4-wire RTD that is being used.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	PT100

Usage

```
[:SENSe[1]]:<function>:RTD:FOUR <type>
[:SENSe[1]]:<function>:RTD:FOUR?
```

`<function>`	The function to which the setting applies; see **Functions**
`<type>`	The type of four-wire RTD: • PT100: `PT100` • PT385: `PT385` • PT3916: `PT3916` • D100: `D100` • F100: `F100` • User-specified type: `USER`

Functions

`VOLTage[:DC]`	`RESistance`	**`TEMPerature`**	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

The transducer type must be set to temperature and the transducer must be set to 4-wire RTD before you can set the RTD type.

Example

`:FUNC "TEMP"`
`:TEMP:TRANsducer FRTD`
`:TEMP:RTD:FOUR PT3916`
	Set the measure function to temperature.
	Set the transducer type to 4-wire RTD.
	Set the RTD type to PT3916.

Also see

[:SENSe[1]]:\<function>:TRANsducer (on page 6-108)

[:SENSe[1]]:<function>:RTD:THRee

This command defines the type of three-wire RTD that is being used.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	PT100

Usage

```
[:SENSe[1]]:<function>:RTD:THRee <type>
[:SENSe[1]]:<function>:RTD:THRee?
```

`<function>`	The function to which the setting applies; see **Functions**
`<type>`	The type of three-wire RTD: • PT100: `PT100` • PT385: `PT385` • PT3916: `PT3916` • D100: `D100` • F100: `F100` • User-specified type: `USER`

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

The transducer type must be set to temperature and the transducer must be set to 3-wire RTD before you can set the RTD type.

Example

`:FUNC "TEMP"`
`:TEMP:TRANsducer TRTD`
`:TEMP:RTD:THR PT3916`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to PT3916.

Also see

[:SENSe[1]]:<function>:TRANsducer (on page 6-108)
Temperature measurements (on page 2-117)

[:SENSe[1]]:<function>:RTD:ZERO

This command contains the zero value of a user-defined RTD.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	100

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	Range: 0 to 10,000

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This attribute is only valid when:

- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

`:FUNC "TEMP"`
`:TEMP:TRANsducer TRTD`
`:TEMP:RTD:THR USER`
`:TEMP:RTD:ALPH 0.00385`
`:TEMP:RTD:ZERO 120`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to User.
	Set the alpha RTD value to 0.00385.
	Set the zero RTD value to 120.

Also see

[:SENSe[1]]:<function>:RTD:THRee (on page 6-97)
[:SENSe[1]]:<function>:TRANsducer (on page 6-108)

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

This command defines the precise acquisition rate at which the digitizing measurements are made.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1,000,000

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	1,000 to 1,000,000 readings per second

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

The sample rate determines how fast the Model DMM7510 acquires a digitized reading.

Set the sample rate before setting the aperture. If the aperture setting is too high for the selected sample rate, it is automatically adjusted to the highest aperture that can be used with the sample rate.

Example

| ```
DIG:FUNC "CURR"
DIG:CURR:SRATE 1000000
DIG:CURR:APER AUTO
DIG:COUN 10
MEAS:DIG?
``` | Set the digitize function to measure current.<br>Set the sample rate to 1,000,000, with a count of 10, and automatic aperture.<br>Make a digitize measurement. |

Also see

[:SENSe[1]]:<function>:APERture (on page 6-49)

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

This command determines if the sense range is set manually or automatically.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<state>`	Disable autorange: `OFF` or `0` Enable autorange: `ON` or `1`

Functions

`VOLTage[:DC]`	`RESistance`	`TEMPerature`	`VOLTage[:DC]:RATio`
`VOLTage:AC`	`FRESistance`	`CONTinuity`	`DIGitize:VOLTage`
`CURRent[:DC]`	`DIODe`	`FREQuency[:VOLTage]`	`DIGitize:CURRent`
`CURRent:AC`	`CAPacitance`	`PERiod[:VOLTage]`	

Details

This selects whether the range for the denominator of the ratio is selected manually or automatically.

This command determines how the range is selected.

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

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

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

Auto range selects the best range in which to measure the signal that is applied to the input terminals of the instrument. When auto range is enabled, the range increases at 120 % of range and decreases occurs when the reading is <10 % of nominal range. For example, if you are on the 1 volt range and auto range is enabled, the instrument auto ranges up to the 10 volt range when the measurement exceeds 1.2 volts. It auto ranges down to the 100 mV range when the measurement falls below 1 volt.

Example

`FUNC "VOLT:RAT"` `VOLT:RAT:SENS:RANG:AUTO OFF`	Set autorange off for the voltage ratio function.

Also see

[:SENSe[1]]:<function>:SENSe:RANGe[:UPPer] (on page 6-101)

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

This command determines the positive full-scale range for the sense measurement.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	Range in volts: `0.1`, `1`, or `10`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

Determines the full-scale input for the reference measurement in the denominator of the ratio. It also affects the accuracy of the measurements and the maximum signal that can be measured. Autorange is automatically set to off if a specific value is set.

When you assign a range value, the instrument is set on a fixed range that is large enough to measure the assigned value. The instrument selects the best range for measuring the maximum expected value.

For example, if you expect a sense reading of approximately 9 V, set the range to 9 V to select the 10 V range.

This command is primarily intended to eliminate the time that is required by the instrument to select an automatic range.

Note that when you select a fixed range, an overflow condition can occur.

When you read this setting, you see the positive full-scale value of the sense range that the instrument is presently using.

Example 1

`:SENS:VOLT:RAT:SENS:RANG 10`	Select the 10 V sense range for DC voltage range.

Also see

Ranges (on page 3-4)
[:SENSe[1]]:<function>:SENSe:RANGe:AUTO (on page 6-100)

[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated

This command sets the simulated reference temperature of the thermocouple reference junction.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	Celsius: 23 Kelvin: 296.15 Fahrenheit: 73.4

Usage

```
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated <tempValue>
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated DEFault
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated MINimum
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated MAXimum
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated?
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated? DEFault
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated? MINimum
[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated? MAXimum
```

`<function>`	The function to which the setting applies; see **Functions**
`<tempValue>`	The temperature: • Celsius: 0 to 65 • Kelvin: 273 to 338 • Fahrenheit: 32 to 149

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This attribute applies to the temperature function when the transducer type is set to thermocouple and the reference junction is set to simulated. It allows you to set the simulated temperature value.

Example

``` TEMP:TRAN TC TEMP:TC:TYPE K TEMP:UNIT C TEMP:TC:RJUN:SIM 30 ```	Sets 30 degrees Celsius as the simulated reference temperature for thermocouples.

Also see

[:SENSe[1]]:<function>:TCouple:TYPE (on page 6-103)
[:SENSe[1]]:<function>:TRANsducer (on page 6-108)
[:SENSe[1]]:<function>:UNIT (on page 6-109)
Temperature measurements (on page 2-117)

[:SENSe[1]]:<function>:TCouple:TYPE

This command indicates the thermocouple type.

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

Usage

```
[:SENSe[1]]:<function>:TCouple:TYPE <identifier>
[:SENSe[1]]:<function>:TCouple:TYPE?
```

`<function>`	The function to which the setting applies; see **Functions**
`<identifier>`	`B`, `E`, `J`, `K`, `N R`, `S`, or `T`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command is only applicable when the transducer type is set to thermocouple.

Example

`TEMP:TRAN TC`
`TEMP:TC:TYPE K`
`TEMP:UNIT C`
`TEMP:TC:RJUN:SIM 30`
	Set the transducer type to thermocouple.
	Set the thermocouple type to K.
	Set the units to Celsius.
	Set the simulated reference temperature to 30.

Also see

[:SENSe[1]]:<function>:TCouple:RJUNction:SIMulated (on page 6-102)
[:SENSe[1]]:<function>:TRANsducer (on page 6-108)
Temperature measurements (on page 2-117)

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

This command describes the type of thermistor.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	5000

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The thermistor type in ohms: • 2252: `2252` • 5000: `5000` • 10000: `10000`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command is only applicable when the transducer type is set to thermistor.

Example

`FUNC "TEMP"`
`TEMP:TRAN THER`
`TEMP:THER 2252`
	Set measurement function to temperature.
	Set the transducer type to thermistor.
	Set the thermistor type to 2252.

Also see

[:SENSe[1]]:<function>:TRANsducer (on page 6-108)
Temperature measurements (on page 2-117)

[:SENSe[1]]:<function>:THReshold:LEVel

This command determines the signal level where the instrument makes frequency or period measurements.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The level: −700 V to 700 V; dependent on range

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

You need to set an appropriate voltage trigger level in order for the frequency counter to operate properly. The frequency counter only counts cycles when the signal amplitude reaches the trigger level. For example, if you set the trigger level for 10 V, any cycles with peak amplitude less than 10 V are not counted.

You must select a specific threshold range (autorange must be set to off) before setting a level that is not zero.

Example

`:FUNC "FREQ"`	Set the measure function to frequency.
`:FREQ:THR:RANG 10`	Set the threshold range to 10 V.
`:FREQ:THR:LEV 5`	Set the threshold level to 5 V.

Also see

[:SENSe[1]]:<function>:THReshold:RANGe (on page 6-106)
[:SENSe[1]]:<function>:THReshold:RANGe:AUTO (on page 6-107)

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

This command indicates the expected input level of the voltage signal.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<n>`	The range: 0.1 to 700; instrument selects nearest valid range (100 mV, 1 V, 10 V, 100 V, 700 V)

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

The range setting conditions the signal. The instrument automatically selects the most sensitive threshold range for the value you enter. For example, if you specify the expected input voltage to be 90 mV, the instrument automatically selects the 100 mV threshold range.

Example

`FREQ:THR:RANG 90`	Set the threshold range for frequency to 90 V, which will select the 100 V range.

Also see

[:SENSe[1]]:<function>:THReshold:LEVel (on page 6-105)
[:SENSe[1]]:<function>:THReshold:RANGe:AUTO (on page 6-107)

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

This command determines if the threshold range is set manually or automatically.

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<state>`	The auto range setting: • Disable: `OFF` or `0` • Enable: `ON` or `1`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

This command determines how the range is selected.

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

When this command is set to on, the instrument uses the signal to determine the most sensitive range on which to perform the measurement. The instrument sets the range when a measurement is requested. To set the range, the instrument makes a measurement to determine the range before making the final measurement, which can result in slower reading times. Turn autorange off and set a specific range to increase measure time.

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

Example

`:FUNC "FREQ"`
`:FREQ:THR:RANG:AUTO OFF`
`:FREQ:THR:RANG 10`
	Set measure function to frequency.
	Disable the threshold autorange.
	Set the range to 10 V.

Also see

[:SENSe[1]]:<function>:THReshold:LEVel (on page 6-105)
[:SENSe[1]]:<function>:THReshold:RANGe (on page 6-106)

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

This command sets the transducer type.

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

Usage

```
[:SENSe[1]]:<function>:TRANsducer <type>
[:SENSe[1]]:<function>:TRANsducer?
```

`<function>`	The function to which the setting applies; see **Functions**
`<type>`	The type of transducer: • Thermocouple: `TCouple` • Thermistor: `THERmistor` • 3-wire RTD: `TRTD` • 4-wire RTD: `FRTD`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

The transducer type determines the type of temperature measurement that is made. Each transducer type has related settings that must also be set. For example, thermocouple measurements are only made if the type is set is set to thermocouple. You also need to set the thermocouple type when setting up a thermocouple. For a transducer type of four-wire RTD, you also set the RTD type.

Example

`:FUNC "TEMP"`	Set the measure function to temperature.
`:TEMP:TRANsducer FRTD`	Set the transducer type to 4-wire RTD.
`:TEMP:RTD:FOUR PT3916`	Set the RTD type to PT3916 for 4-wire RTDs.

Also see

[:SENSe[1]]:<function>:RTD:FOUR (on page 6-96)
[:SENSe[1]]:<function>:RTD:THRee (on page 6-97)
[:SENSe[1]]:<function>:TCouple:TYPE (on page 6-103)
[:SENSe[1]]:<function>:THERmistor (on page 6-104)
Temperature measurements (on page 2-117)

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

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

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

Usage

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

`<function>`	The function to which the setting applies; see **Functions**
`<unitOfMeasure>`	Temperature: `KELVin`, `CELSius`, or `FAHRenheit` Digitize voltage, AC voltage, and DC voltage: `VOLT` or `DB`

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

The change in measurement units is displayed when the next measurement occurs. You can only change the units for the voltage, temperature, and digitize voltage functions. Other functions have a fixed units setting that cannot be changed.

Example

`VOLT:UNIT DB`	Changes the front-panel display and buffer readings for DC voltage measurements to be displayed in decibels.

Also see

None

[:SENSe[1]]:AZERo:ONCE

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

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

Usage

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

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

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

Example

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

Also see

Automatic reference measurements (on page 2-147)
[:SENSe[1]]:<function>:AZERo[:STATe] (on page 6-68)

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

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

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

Usage

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

Details

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

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

Example

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

Also see

Configuration lists (on page 3-32)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-112)

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

This command creates an empty measure configuration list.

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

Usage

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

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

Details

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

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

Example

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

Creates a measure configuration list named `MyMeasList`.

Also see

*SAV (on page 6-12)
Configuration lists (on page 3-32)
[:SENSe[1]]:CONFiguration:LIST:STORe (on page 6-115)

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

This command deletes a measure configuration list.

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

Usage

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

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

Details

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

Example

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

Also see

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

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

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

Usage

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

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

Details

This command returns data for one configuration point.

Example

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

Returns the TSP commands and parameter settings that represent the settings in configuration point 2.
Example partial output:
```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.unit = dmm.UNIT_VOLT
dmm.measure.range = 1
dmm.measure.autorange = dmm.ON
dmm.measure.autozero.enable = dmm.ON
```

Also see

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

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

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

Usage

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

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

Details

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

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

This command returns data for one configuration point.

Example

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

Also see

Configuration lists (on page 3-32)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-112)
[:SENSe[1]]:CONFiguration:LIST:STORe (on page 6-115)

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

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

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

Usage

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

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

Details

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

Example

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

Also see

Configuration lists (on page 3-32)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-112)

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

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

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

Usage

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

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

Details

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

Example

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

Also see

Configuration lists (on page 3-32)
[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-112)

[:SENSe[1]]:COUNt

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

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

Usage

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

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

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATIO
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

This command sets the count for all measure functions.

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

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

Example

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

Also see

[:SENSe[1]]:DIGitize:COUNt

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

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	10,000

Usage

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

`<n>`	The number of measurements (1 to 55,000,000)

Details

The digitizer makes the number of readings set by this command in the time set by the sample rate. This command does not affect the trigger model.

Example

`DIG:COUN 10`	Make ten measurements.

Also see

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

This command selects which digitize function is active.

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

Usage

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

<function>	A string that contains the measurement function to make active: • Current: CURRent • Voltage: VOLTage

Details

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

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

If you send the query when a measurement function is selected, the query returns NONE.

If a basic (non-digitize) measurement function is selected, this returns NONE. The none setting is automatically made if you select a function with [:SENSe[1]]:FUNCtion[:ON] or through the options from the front-panel Measure Functions tab.

Example

DIG:FUNC "VOLTage"	Make the digitize voltage function the active function.

Also see

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

This command selects the active measure function.

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

Usage

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

<function>	A string that contains the measure function; see **Functions**

Functions

VOLTage[:DC]	RESistance	TEMPerature	VOLTage[:DC]:RATio
VOLTage:AC	FRESistance	CONTinuity	DIGitize:VOLTage
CURRent[:DC]	DIODe	FREQuency[:VOLTage]	DIGitize:CURRent
CURRent:AC	CAPacitance	PERiod[:VOLTage]	

Details

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

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

If you send this query when a digitize measurement function is selected, this returns NONE.

Example

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

Also see

Making resistance measurements
[:SENSe[1]]:DIGitize:FUNCtion[:ON] (on page 6-118)

[:SENSe[1]]:TRIGger:DIGitize:STIMulus

This command sets the instrument to digitize a measurement the next time it detects the specified trigger event.

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

Usage

```
[:SENSe[1]]:TRIGger:DIGitize:STIMulus <event>
[:SENSe[1]]:TRIGger:DIGitize:STIMulus?
```

<event>	The event to use as a stimulus; see **Details**

Details

This command is intended to provide the lowest possible latency between a trigger event such as digital I/O and a reading. It forces the instrument to make a digitize measurement the next time it detects the specified trigger event. Options for the trigger event parameter are listed in the following table.

A digitize function must be active before sending this command. The measurement is digitized for the active function. If a measure function is active, an error is generated.

Before using this command, set the active reading buffer. Readings are stored in the active reading buffer.

If the count is set to more than 1, the first reading is initialized by this trigger. Subsequent readings occur as rapidly as the instrument can make them. If a trigger occurs during the group measurement, the trigger is latched and another group of measurements with the same count will be triggered after the current group completes.

If the stimulus is set to none, this command has no effect on readings.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

```*RST```
 ```:DIG:FUNC "VOLT"```
 ```:TRIG:DIG:STIM DISP```
 ```:DIG:COUN 10```
	Reset the instrument.
	Set the digitize function to voltage.
	Set the digitize trigger stimulus to be the front-panel TRIGGER key.
	Set the digitize count to 10.
	Press the TRIGGER key to generate 10 readings.

Also see

[:SENSe[1]]:TRIGger:MEASure:STIMulus

This command sets the instrument to make a measurement the next time it detects the specified trigger event.

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

Usage

```
[:SENSe[1]]:TRIGger:MEASure:STIMulus <event>
[:SENSe[1]]:TRIGger:MEASure:STIMulus?
```

`<event>`	The event to use as a stimulus; see **Details**

Details

This command is intended to provide the lowest possible latency between an event such as digital I/O and a reading. It forces the instrument to make a measurement the next time it detects the specified trigger event. Options for the trigger event parameter are listed in the following table.

A measure function must be active before sending this command. The measurement is made for the active measure function. If a digitize function is active, an error is generated.

Before using this command, set the active reading buffer. Readings are stored in the active reading buffer.

If the count is set to more than 1, the first reading is initialized by this trigger. Subsequent readings occur as rapidly as the instrument can make them. If a trigger occurs during the group measurement, the trigger is latched and another group of measurements with the same count will be triggered after the current group completes.

If the stimulus is set to none, this command has no effect on readings.

Trigger events	
Event description	**Event constant**
No trigger event	NONE
Front-panel TRIGGER key press	DISPlay
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	NOTify`<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	COMMand
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	DIGio`<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	TSPLink`<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	LAN`<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	BLENder`<n>`
Trigger timer `<n>` (1 to 4) expired	TIMer`<n>`
Analog trigger	ATRigger
External in trigger	EXTernal

Example

`*RST`
`:FUNC "CURR"`
`:TRIG:MEAS:STIM DISP`
`:COUN 10`
	Reset the instrument.
	Set the function to DC current.
	Set the trigger stimulus to be the front-panel TRIGGER key.
	Set the count to 10.
	Press the TRIGGER key to generate 10 readings.

Also see

:READ? (on page 6-7)
[:SENSe[1]]:FUNCtion[:ON] (on page 6-118)

STATus subsystem

The STATus subsystem controls the status registers of the instrument. For additional information on the status model, see Status model (on page C-1).

:STATus:CLEar

This function clears event registers and the event log.

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

Usage

```
:STATus:CLEar
```

Details

This command clears the event registers of the Questionable Event and Operation Event Register set. It does not affect the Questionable Event Enable or Operation Event Enable registers.

Example

:STATus:CLEar	Clear the bits in the registers

Also see

*CLS (on page B-2)

:STATus:OPERation:CONDition?

This command reads the Operation Event Register of the status model.

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

Usage

```
:STATus:OPERation:CONDition?
```

Details

This command reads the contents of the Operation Condition Register, which is one of the Operation Event Registers.
For detail on interpreting the value of a register, see Understanding bit settings (on page C-14).

Example

:STAT:OPER:COND?	Returns the contents of the Operation Condition Register.

Also see

Operation Event Register (on page C-7)

:STATus:OPERation:ENABle

This command sets or reads the contents of the Operation Event Enable Register of the status model.

Type	Affected by	Where saved	Default value
Command and query	STATus:PRESet	Not applicable	0

Usage

```
:STATus:OPERation:ENABle <n>
:STATus:OPERation:ENABle?
```

<n>	The status of the operation status register

Details

This command sets or reads the contents of the Enable register of the Operation Event Register.

When one of these bits is set, when the corresponding bit in the Operation Event Register or Operation Condition Register is set, the OSB bit in the Status Byte Register is set.

When sending binary values, preface `<n>` with `#b`. When sending hexadecimal values, preface `<n>` with `#h`. No preface is needed when sending decimal values.

Example

`:STAT:OPER:ENAB #b0101000000000000`	Sets the 12 and 14 bits of the operation status enable register using a decimal value.
	You could also send the decimal value:
	`:STAT:OPER:ENAB 20480`
	Or the hexadecimal value:
	`:STAT:OPER:ENAB #h5000`

Also see

Operation Event Register (on page C-7)

:STATus:OPERation:MAP

This command allows you to map event numbers to bits in the Operation Event Registers.

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

Usage

```
:STATus:OPERation:MAP <bitNumber>, <setEvent>
:STATus:OPERation:MAP <bitNumber>, <setEvent>, <clearEvent>
:STATus:OPERation:MAP? <bitNumber>
```

bitNumber	The bit number that is mapped to an event (0 to 14)
setEvent	The number of the event that sets the bits in the condition and event registers; 0 if no mapping
clearEvent	The number of the event that clears the bit in the condition register; 0 if no mapping

Details

You can map events to bits in the event registers with this command. This allows you to cause bits in the condition and event registers to be set or cleared when the specified events occur. You can use any valid event number as the event that sets or clears bits.

When a mapped event is programmed to set bits, the corresponding bits in both the condition register and event register are set when the event is detected.

When a mapped event is programmed to clear bits, the bit in the condition register is set to 0 when the event is detected.

If the event is set to zero (0), the bit is never set.

The query requests the mapped set event and mapped clear event status for a bit in the Operation Event Registers. When you query the mapping for a specific bit, the instrument returns the events that were mapped to set and clear that bit. Zero (0) indicates that the bits have not been set.

Example

:STATus:OPERation:MAP 0, 4916, 4917	When event 4916 (the buffer is 0 % filled) occurs, bit 0 is set in the condition register and the event register of the Operation Event Register. When event 4917 (buffer is 100 % filled) occurs, bit 0 in the condition register is cleared.

Also see

Programmable status register sets (on page C-5)

:STATus:OPERation[:EVENt]?

This command reads the Operation Event Register of the status model.

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

Usage

```
:STATus:OPERation[:EVENt]?
```

Details

This attribute reads the operation event register of the status model.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Example

stat:oper?	Returns the contents of the Operation Event Register of the status model.

Also see

Operation Event Register (on page C-7)

:STATus:PRESet

This command resets all bits in the status model.

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

Usage

```
:STATus:PRESet
```

Details

This function clears the event registers and the enable registers for operation and questionable. It will not clear the Service Request Enable Register (*SRE) to Standard Request Enable Register (*ESE).

Preset does not affect the event queue.

The Standard Event Status Register is not affected by this command.

Example

STAT:PRES	Resets the registers.

Also see

Status model (on page C-1)

:STATus:QUEStionable:CONDition?

This command reads the Questionable Condition Register of the status model.

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

Usage

```
:STATus:QUEStionable:CONDition?
```

Details

This command reads the contents of the Questionable Condition Register, which is one of the Questionable Event Registers.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-14).

Example

:STAT:QUES:COND?	Reads the Questionable Condition Register.

Also see

Questionable Event Register (on page C-7)
Understanding bit settings (on page C-14)

:STATus:QUEStionable:ENABle

This command sets or reads the contents of the questionable event enable register of the status model.

Type	Affected by	Where saved	Default value
Command and query	STATus:PRESet	Not applicable	0

Usage

```
:STATus:QUEStionable:ENABle <n>
:STATus:QUEStionable:ENABle?
```

<n>	The value of the register (0 to 65535)

Details

This command sets or reads the contents of the Enable register of the Questionable Event Register.

When one of these bits is set, when the corresponding bit in the Questionable Event Register or Questionable Condition Register is set, the MSB and QSM bits in the Status Byte Register are set.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-14).

Example 1

:STAT:QUES:ENAB 8 :STAT:QUES:ENAB?	Enable bit 4, Limit 3 Fail, when the limit test 3 failure value is exceeded. Check to see that the value was set.

Also see

None

:STATus:QUEStionable:MAP

This command queries mapped event numbers or maps event numbers to bits in the event registers.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Not applicable	0

Usage

```
:STATus:QUEStionable:MAP <bitNumber>, <setEvent>
:STATus:QUEStionable:MAP <bitNumber>, <setEvent>, <clearEvent>
:STATus:QUEStionable:MAP? <bitNumber>
```

`<bitNumber>`	The bit number that is mapped to an event (0 to 14)
`<setEvent>`	The number of the event that sets the bits in the condition and event registers; 0 if no mapping
`<clearEvent>`	The number of the event that clears the bit in the condition register; 0 if no mapping

Details

You can map events to bits in the event registers with this command. This allows you to cause bits in the condition and event registers to be set or cleared when the specified events occur. You can use any valid event number as the event that sets or clears bits.

When a mapped event is programmed to set bits, the corresponding bits in both the condition register and event register are set when the event is detected.

When a mapped event is programmed to clear bits, the bit in the condition register is set to 0 when the event is detected.

If the event is set to zero (0), the bit is never set.

When you query the mapping for a specific bit, the instrument returns the events that were mapped to set and clear that bit. Zero (0) indicates that the bits have not been set.

Example

`:STAT:QUES:MAP 0, 4916, 4917`	When event 4916 (the buffer is 0 % filled) occurs, bit 0 is set in the condition register and the event register of the Questionable Event Register. When event 4917 (buffer is 100 % filled) occurs, bit 0 in the condition register is cleared.

Also see

None

:STATus:QUEStionable[:EVENt]?

This command reads the Questionable Event Register.

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

Usage

```
:STATus:QUEStionable[:EVENt]?
```

Details

This query reads the contents of the questionable status event register. After sending this command and addressing the instrument to talk, a value is sent to the computer. This value indicates which bits in the appropriate register are set.

The Questionable Register can be set to the numeric equivalent of the bit to set. To set more than one bit of the register, set the Questionable Register to the sum of their decimal weights. For example, to set bits B12 and B13, set the Questionable Register to 12,288 (which is the sum of 4,096 + 8,192).

Example

`:STAT:QUES?`	Query the Questionable Register.

Also see

Questionable Event Register (on page C-7)

SYSTem subsystem

This subsystem contains commands that affect the overall operation of the instrument, such as passwords, beepers, communications, event logs, and time.

:SYSTem:ACCess

This command contains the type of access users have to the instrument through different interfaces.

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

Usage

```
:SYSTem:ACCess <permissions>
:SYSTem:ACCess?
```

`<permissions>`	The level of access that is allowed: • Full access for all users from all interfaces: `FULL` • Allows access by one remote interface at a time with login and logout required from other interfaces: `EXCLusive` • Allows access by one remote interface at a time with passwords required on all interfaces: `PROTected` • Allows access by one interface at a time (including the front panel) with passwords required on all interfaces: `LOCKout`

Details

When access is set to full, the instrument accepts commands from any interface with no login or password.

When access is set to exclusive, you must log out of one remote interface and log into another one to change interfaces. You do not need a password with this access.

Protected access is similar to exclusive access, except that you must enter a password when logging in.

When the access is set to locked out, a password is required to change interfaces, including the front panel interface.

Under any access type, if a script is running on one remote interface when a command comes in from another remote interface, the command is ignored and the message "FAILURE: A script is running, use ABORT to stop it" is generated.

Example

`:SYST:ACC LOCK`
`login admin`
`logout`
	Set the instrument access to locked out.
	Log into the interface using the default password.
	Log out of the interface.

Also see

[:SYSTem:PASSword:NEW](#) (on page 6-141)

:SYSTem:BEEPer[:IMMediate]

This command generates an audible tone.

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

Usage

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

`<frequency>`	The frequency of the beep (20 to 8000 Hz)
`<duration>`	The amount of time to play the tone (0.001 to 100 s)

Details

You can use the beeper of the instrument to provide an audible signal at a specific frequency and time duration.

Example

`:SYSTem:BEEPer 500, 1`	Beep at 500 Hz for 1 s.

Also see

None

:SYSTem:CLEar

This command clears the event log.

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

Usage

```
:SYSTem:CLEar
```

Details

This command removes all events from the event log, including entries in the front-panel event log.

Also see

:SYSTem:COMMunication:LAN:CONFigure

This command specifies the LAN configuration for the instrument.

Type	Affected by	Where saved	Default value
Command and query	Rear panel LAN reset	Nonvolatile memory	AUTO

Usage

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

AUTO	Use automatically configured LAN settings (default)
MANual	Use manually configured LAN settings
<IPaddress>	LAN IP address; must be a string specifying the IP address in dotted decimal notation; required if the mode is set to manual (default "0.0.0.0")
<NETmask>	The LAN subnet mask; must be a string in dotted decimal notation (default "255.255.255.0")
<GATeway>	The LAN default gateway; must be a string in dotted decimal notation (default "0.0.0.0")

Details

This command specifies how the LAN IP address and other LAN settings are assigned. If automatic configuration is selected, the instrument automatically determines the LAN information. When method is automatic, the instrument first attempts to configure the LAN settings using dynamic host configuration protocol (DHCP). If DHCP fails, it tries dynamic link local addressing (DLLA). If DLLA fails, an error occurs.

If manual is selected, you must define the IP address. You can also assign a subnet mask, and default gateway. The IP address, subnet mask, and default gateway must be formatted in four groups of numbers, each separated by a decimal. If you do not specify a subnet mask or default gateway, the previous settings are used. When specifying multiple parameters, do not use spaces after the commas.

The query form of the command returns the present settings in the order shown here.

Automatic:

```
AUTO,<IPaddress>,<NETmask>,<GATeway>
```

Manual:

```
MANual,<IPaddress>,<NETmask>,<GATeway>
```

Example

```
SYST:COMM:LAN:CONF "MANUAL,192.168.0.1,255.255.240.0,192.168.0.3"
SYST:COMM:LAN:CONF?
```

Set the IP address to be set manually, with the IP address set to 192.168.0.1, the subnet mask to 255.255.240.0, and the gateway address to 192.168.0.3.
Query to verify the settings. The response to the query should be:

```
manual,192.168.0.1,255.255.240.0,192.168.0.3
```

Also see

:SYSTem:COMMunication:LAN:MACaddress? (on page 6-133)

:SYSTem:COMMunication:LAN:MACaddress?

This command queries the LAN MAC address.

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

Usage

```
:SYSTem:COMMunication:LAN:MACaddress?
```

Details

The MAC address is a character string representing the MAC address of the instrument in hexadecimal notation. The string includes colons that separate the address octets.

Example

:SYSTem:COMMunication:LAN:MACaddress?	Returns the MAC address. For example, you might see: `08:00:11:00:00:57`

Also see

:SYSTem:ERRor[:NEXT]?

This command returns the oldest unread error message from the event log and removes it from the log.

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

Usage

```
:SYSTem:ERRor[:NEXT]?
```

Details

As error and status messages occur, they are placed in the event log. The event log is a first-in, first-out (FIFO) register that can hold up to 1000 messages.

This command returns the next entry from the event log.

This command does not affect the event log that is displayed on the front panel.

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

```
0,"No error;0,0,0"
```

This command returns only error messages from the event log. To return information and warning messages, see `:SYSTem:EVENtlog:NEXT?`.

Note that if you have used `:SYSTem:ERRor[:NEXT]?` to check events, `:SYSTem:EVENtlog:NEXT?` shows the next event item after the last error that was returned by `:SYSTem:ERRor[:NEXT]?` You will not see warnings or information event log items that occurred before you used `:SYSTem:ERRor[:NEXT]?`

Example

`SYST:ERR:NEXT?`	Returns information on the next error in the event log. For example, if you sent a command without a parameter, the return is: `-109,"Missing parameter;1,1367794863,384492889"`

Also see

:SYSTem:ERRor:CODE[:NEXT]?

This command reads the oldest error code.

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

Usage

```
:SYSTem:ERRor:CODE[:NEXT]?
```

Details

This command returns the numeric code of the next error in the event log. The error is cleared from the queue after being read.

This command returns only error messages from the event log. To return information and warning messages, see `:SYSTem:EVENtlog:NEXT?`

Example

`SYST:ERR:CODE?`	Returns the error code of the next error in the event log. For example, if error -222, Parameter data out of range error, occurred, the output is: `-222`

Also see

:SYSTem:EVENtlog:NEXT? (on page 6-137)

:SYSTem:ERRor:COUNt?

This command returns the number of errors in the event log.

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

Usage

```
:SYSTem:ERRor:COUNt?
```

Details

This command does not return other types of events, such as information messages. To return other types of events, use `:SYSTem:EVENtlog:COUNt?`

This command does not clear the errors from the event log.

Example

`SYST:ERR:COUN?`	If there are five errors in the event log, the output is: `5`

Also see

:SYSTem:EVENtlog:COUNt? (on page 6-136)

:SYSTem:EVENtlog:COUNt?

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

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

Usage

```
:SYSTem:EVENtlog:COUNt?
:SYSTem:EVENtlog:COUNt? <eventType>
:SYSTem:EVENtlog:COUNt? <eventType>, <eventType>
```

`<eventType>`	Limits the list of event log entries to specific types; set to: - Returns the number of errors: `ERRor` - Returns the number of warnings: `WARNing` - Returns the number of informational messages: `INFormational` - Returns all events: `ALL`

Details

A count finds the number of unread events in the event log. You can specify the event types to return, or return the count for all events.

This command reports the number of events that have occurred since the command was last sent and or since the event log was last cleared.

Example

`:SYST:EVEN:COUN? ERR`	Displays the present number of errors in the instrument event log. If there are three errors in the event log, output is: `3`

Also see

[:SYSTem:CLEar](#) (on page 6-131)
[:SYSTem:EVENtlog:NEXT?](#) (on page 6-137)
[:SYSTem:EVENtlog:SAVE](#) (on page 6-139)

:SYSTem:EVENtlog:NEXT?

This command returns the oldest unread event message from the event log.

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

Usage

```
:SYSTem:EVENtlog:NEXT?
:SYSTem:EVENtlog:NEXT? <eventType>
:SYSTem:EVENtlog:NEXT? <eventType>, <eventType>
:SYSTem:EVENtlog:NEXT? <eventType>, <eventType>, <eventType>
```

`<eventType>`	Limits the event log entries that are returned to specific types; set to:
	• Returns only the next error: `ERRor`
	• Returns only the next warning: `WARNing`
	• Returns only the next informational message: `INFormational`
	• Returns any event: `ALL`

Details

When an event occurs on the instrument, it is placed in the event log. The `:SYSTem:EVENtlog:NEXT?` command retrieves an unread event from the event log. Once an event is read, it can no longer be accessed remotely. However, it can be viewed on the front panel.

To read multiple commands, execute this command multiple times.

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

`0,"No error;0,0,0"`

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

Note that if you have used `:SYSTem:ERRor[:NEXT]?` to check events, `:SYSTem:EVENtlog:NEXT?` shows the next event item after the last error that was returned by `:SYSTem:ERRor[:NEXT]?` You will not see warnings or information event log items that occurred before you used `:SYSTem:ERRor[:NEXT]?`

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

The information that is returned is in the order:

`<eventNumber>, <message>, <eventType>, <timeSeconds>, <timeNanoSeconds>`

`<eventNumber>`	The event number
`<message>`	A description of the event
`<eventType>`	The type of event:
	• Error only: 1
	• Warning only: 2
	• Information only: 4
`<timeSeconds>`	The seconds portion of the time when the event occurred
`<timeNanoSeconds>`	The fractional seconds portion of the time when the event occurred

Example

`SYST:EVEN:NEXT?`	Returns information on the next event in the event log. For example, if you sent a command without a parameter, the return is: `-109,"Missing parameter;1,1367794863,384492889"`

Also see

[:SYSTem:CLEar](on page 6-131)

:SYSTem:EVENtlog:POST

This command allows you to post your own text to the event log.

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

Usage

```
:SYSTem:EVENtlog:POST "<message>"
:SYSTem:EVENtlog:POST "<message>", <eventType>
```

`<message>`	A string that contains the message that will be associated with this event
`<eventType>`	The type of event that is generated; set to: • The error type: `ERRor` • The warning type: `WARNing` • The informational type: `INFormational` (default)

Details

You can use this command to create your own event log entries and assign a severity level to them. This can be useful for debugging and status reporting.

From the front panel, you must set the Log Warnings and Log Information options on to have the custom warning and information events placed into the event log.

Example

`*CLS`
 `SYST:EVEN:POST "my error", INF`
 `SYST:EVEN:NEXT?`
	Clear the event log.
	 Post an error named `my error`.
	 Output:
	 `1003,"User: my error;4,1400469179,431599191"`

Also see

None

:SYSTem:EVENtlog:SAVE

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

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

Usage

```
:SYSTem:EVENtlog:SAVE "<filename>"
:SYSTem:EVENtlog:SAVE "<filename>", <eventType>
```

`<filename>`	A string that holds the name of the file to be saved
`<eventType>`	Limits the event log entries that are saved to specific types; set to: • `ERRor`: Saves only error entries • `WARNing`: Saves only warning entries • `INFormational`: Saves only informational messages • `ALL`: Saves all event log entries (default)

Details

This command saves all event log entries to a USB flash drive.

If you do not define an event type, the instrument saves all event log entries.

The extension `.csv` is automatically added to the file name.

Example

`SYST:EVEN:SAVE "/usb1/July_error_log", ERR`	Saves the error events in the event log to a file on the USB flash drive named `July_error_log.csv`.

Also see

[:SYSTem:CLEar](#) (on page 6-131)

:SYSTem:GPIB:ADDRess

This command contains the GPIB address.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	16

Usage

```
:SYSTem:GPIB:ADDRess <n>
:SYSTem:GPIB:ADDRess?
```

<n>	The GPIB address of the instrument (1 to 30)

Details

The address can be set to any address value from 1 to 30. However, the address must be unique in the system. It cannot conflict with an address that is assigned to another instrument or to the GPIB controller.

A new GPIB address takes effect when the command to change it is processed. If there are response messages in the output queue when this command is processed, they must be read at the new address.

If command messages are being queued (sent before this command has executed), the new settings may take effect in the middle of a subsequent command message, so care should be exercised when setting this attribute from the GPIB interface.

You should allow ample time for the command to be processed before attempting to communicate with the instrument again. After sending this command, make sure to use the new address to communicate with the instrument.

`*RST` does not affect the GPIB address.

Example

`:SYSTem:GPIB:ADDRess 26` `:SYSTem:GPIB:ADDRess?`	Sets the GPIB address and reads the address. Output: `2.600000e+01`

Also see

:SYSTem:LFRequency?

This query contains the power line frequency setting that is used for NPLC calculations.

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

Usage

```
:SYSTem:LFRequency?
```

Details

The instrument automatically detects the power line frequency (either 50 Hz or 60 Hz) when the instrument is powered on.

Example

`:SYST:LFR?`	Check the line frequency.

Also see

None

:SYSTem:PASSword:NEW

This command stores the instrument password.

Type	Affected by	Where saved	Default value
Command only	Rear-panel LAN reset	Nonvolatile memory	admin

Usage

```
:SYSTem:PASSword:NEW "<password>"
```

`<password>`	A string that contains the instrument password (maximum 30 characters)

Details

When the access to the instrument is set to protected or lockout, this is the password that is used to gain access.

If you forget the password, you can reset the password to the default:

1. On the front panel, press **MENU**.
2. Under System, select **Info/Manage**.
3. Select **Password Reset**.

You can also reset the password and the LAN settings from the rear panel by inserting a straightened paper clip into hole below LAN RESET.

Example

`SYST:PASS:NEW "N3wpa55w0rd"`	Change the password of the instrument to `N3wpa55w0rd`.

Also see

[:SYSTem:ACCess](#) (on page 6-130)

:SYSTem:POSetup

This command selects the defaults that are used when you power on the instrument.

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

Usage

```
:SYSTem:POSetup <name>
:SYSTem:POSetup?
```

`<name>`	Which setup to restore when you power on the instrument:
	• Power on to `*RST` defaults: `RST`
	• Stored setup 0: `SAV0`
	• Stored setup 1: `SAV1`
	• Stored setup 2: `SAV2`
	• Stored setup 3: `SAV3`
	• Stored setup 4: `SAV4`

Details

When you select `RST`, the instrument restores settings to their default values when the instrument is powered on.

When you select a `SAV` option, the settings in the selected saved setup are applied when the instrument is powered on. The settings are saved using the `*SAV` command.

Example

`SYST:POS SAV1`	Set the instrument to restore the settings that are saved in the stored setup 1 when the instrument is powered on.

Also see

*SAV (on page 6-12)

:SYSTem:TEMPerature:INTernal?

This command returns the internal temperature of the instrument.

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

Usage

```
:SYSTem:TEMPerature:INTernal?
```

Details

Returns the last recorded value of internal temperature of the instrument in Celsius (°C). The instrument checks internal temperature when it updates references when autozero is on. Internal temperature is not checked if autozero is set to off. It can also become stale when digitize measurements are used or when measurements take a long time.

If the temperature changes more than ±5 °C, the instrument logs an event and displays a message on the front panel that recommends that you perform auto calibration.

Example

SYST:TEMP:INT?	Returns the internal temperature of the instrument. Example output: 53.732574528

Also see

:ACAL:LASTrun:TEMPerature:INTernal? (on page 6-14)
:ACAL:RUN (on page 6-18)

:SYSTem:TIME

This command sets the absolute time of the instrument.

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

Usage

```
:SYSTem:TIME <year>, <month>, <day>, <hour>, <minute>, <second>
:SYSTem:TIME <hour>, <minute>, <second>
:SYSTem:TIME?
:SYSTem:TIME? 1
```

`<year>`	Year; must be more than 1970
`<month>`	Month (1 to 12)
`<day>`	Day (1 to 31)
`<hour>`	Hour in 24-hour time format (0 to 23)
`<minute>`	Minute (0 to 59)
`<second>`	Second (0 to 59)

Details

When queried without a parameter, this command returns the present timestamp value in seconds since January 1, 1970 to the nearest second.

If you query with `1`, this command returns the present timestamp in the format:

```
<weekday> <month> <day> <hour>:<minute>:<second> <year>
```

Where `<weekday>` is the day of the week.

Internally, the instrument bases time in UTC time. UTC time is specified as the number of seconds since Jan 1, 1970, UTC. You can use UTC time from a local time specification, or you can use UTC time from another source (for example, your computer).

Example

`syst:time 2014, 8, 29, 11, 30, 30` `syst:time? 1`	Set the system time to August 29, 2014 at 11:30:30 and confirm setting. Output: `Fri Aug 29 11:30:37 2014`

Also see

None

:SYSTem:VERSion?

Query the present SCPI version.

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

Usage

```
:SYSTem:VERSion?
```

Details

This query command returns the SCPI version.

Example

`SYSTem:VERSion?`	Query the version. An example of a return is: `1996.0`

Also see

None

TRACe subsystem

The TRACe subsystem contains commands that control the reading buffers.

:TRACe:ACTual?

This command contains the number of readings in the specified reading buffer.

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

Usage

```
:TRACe:ACTual?
:TRACe:ACTual? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This command returns the number of readings stored in the buffer.

Example

`TRACe:MAKE "testData", 200`
`COUN 10`
`MEASure:CURRent? "testData"`
	Creates 200 element reading buffer named `testData`.
	Set the measurement count to 10.
	Set the measurement function to current. Make readings, and store the readings in `testData`. Returns the 10th measurement reading after taking all 10 readings.

`:TRACe:ACTual?`	Returns the number of readings in `defbuffer1`. Example output: `850`
`:TRACe:ACTual? "testData"`	Returns the number of readings in the buffer `testData`. Example output: `10`

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)

:TRACe:CLEar

This command clears all readings and statistics from the specified buffer.

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

Usage

```
:TRACe:CLEar
:TRACe:CLEar "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:MAKE "testData", 200`
`MEASure:RESistance? "testData"`
`TRACe:ACTual? "testData"`
`TRACe:CLEar "testData"`
`TRACe:ACTual? "testData"`
	Create user-defined buffer named `testData`.
	Take a measurement and store it in `testData` and returns the last reading measured.
	Verify that there is data in `testData` buffer.
	Output:
	`1`
	Clear `testData` buffer.
	Verify that `testData` is empty.
	Output:
	`0`

`TRACe:CLEar`
`TRACe:CLEar "defbuffer1"`
`TRACe:CLEar "defbuffer2"`
	Clear the default buffer. This command clears `defbuffer1`.
	Clear `defbuffer1`. Specify default buffer by name.
	Clear `defbuffer2`. Specify default buffer by name.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)

:TRACe:DATA?

This command returns specified data elements from a specified reading buffer.

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

Usage

```
:TRACe:DATA? <startIndex>, <endIndex>
:TRACe:DATA? <startIndex>, <endIndex>, "<bufferName>"
:TRACe:DATA? <startIndex>, <endIndex>, "<bufferName>", <bufferElements>
:TRACe:DATA? <startIndex>, <endIndex>, "<bufferName>", <status>
```

`<startIndex>`	Beginning index of the buffer to return
`<endIndex>`	Ending index of the buffer to return
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<bufferElements>`	A list of elements in the buffer to print; if nothing is specified, `READing` is used; see **Details** for the list of options
`<status>`	Only available if `:FORMat[:DATA]` is set to `ASCii`; returns buffer status bits; see **Details** for the bit definitions

Details

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

When specifying buffer elements, you can:

- Specify buffer elements in any order.

- Include up to 12 elements in a single list. You can repeat elements as long as the number of elements in the list is less than 12.

- Use a comma to delineate multiple elements for a data point.

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

Option	Description
DATE	The date when the data point was measured; not available for reading buffers that are set to the style compact
EXTRa	Returns an additional value (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option
FORMatted	The measured value as it appears on the front panel
FRACtional	The fractional seconds when the data point was measured
READing	The measurement reading
RELative	The relative time when the data point was measured
SEConds	The seconds in UTC (Coordinated Universal Time) format when the data point was measured
STATus	The status information associated with the measurement
TIME	The time when the data point was measured
TSTamp	The timestamp when the data point was measured
UNIT	The unit of measure of the measurement

The `<status>` parameter returns status values for the readings in the buffer. The status values are floating-point numbers that encode the status value. Refer to the following table for values.

Buffer status bits for sense measurements

Bit (hex)	Name	Decimal	Description
0x0001	STAT_QUESTIONABLE	1	Measure status questionable
0x0006	STAT_ORIGIN	6	A/D converter from which reading originated; for the Model DMM7510, this will always be 0 (Main) or 2 (digitizer)
0x0008	STAT_TERMINAL	8	Measure terminal, front is 1, rear is 0
0x0010	STAT_LIMIT2_LOW	16	Measure status limit 2 low
0x0020	STAT_LIMIT2_HIGH	32	Measure status limit 2 high
0x0040	STAT_LIMIT1_LOW	64	Measure status limit 1 low
0x0080	STAT_LIMIT1_HIGH	128	Measure status limit 1 high
0x0100	STAT_START_GROUP	256	First reading in a group

Example

`TRAC:MAKE "buf100", 100`
`TRIGger:LOAD "SimpleLoop", 5, 0, "buf100"`
`INIT`
`*WAI`
`TRAC:DATA? 1, 5, "buf100", READ, REL`
`TRAC:DATA? 1, 5, "buf100", REL`
`TRAC:DATA? 1, 3, "buf100"`
	Create a buffer called `buf100` with a maximum size of 100.
	Set the instrument to configure the trigger model to loop, taking 5 readings with no delay, and store the readings in the `buf100` reading buffer.
	Initiate the trigger model and wait for the trigger model to complete. The trigger model will make 5 readings and store them in `buf100`.
	Read 5 data points and include the reading and relative time for each data point.
	Output:
	`5.043029E-05,0.000000,5.016920E-`
	`    05,0.020199,5.047250E-`
	`    05,0.040201,5.001598E-`
	`    05,0.079671,5.053504E-05,0.099205`
	Read 5 data points and include relative time for each data point.
	Output:
	`0,0.020199,0.040201,0.079671,0.099205`
	Returns the first 3 reading values from `buf100` reading buffer.
	Output:
	`5.043029E-05,5.016920E-05,5.047250E-05`

Also see

:FORMat[:DATA] (on page 6-46)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)

:TRACe:DELete

This command deletes a user-defined reading buffer.

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

Usage

`:TRACe:DELete "<bufferName>"`

`<bufferName>`	A string that contains the name of the user-defined reading buffer to delete

Details

You cannot delete the default reading buffers, `defbuffer1` and `defbuffer2`.

Example

`TRAC:DEL "testData"`	Delete the `testData` buffer.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)

:TRACe:FILL:MODE

This command determines if a reading buffer is filled continuously or is filled once and stops.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	defbuffer1: CONT defbuffer2: CONT User-defined buffers: ONCE

Usage

```
:TRACe:FILL:MODE <fillType>
:TRACe:FILL:MODE <fillType>, "<bufferName>"
:TRACe:FILL:MODE?
:TRACe:FILL:MODE? "<bufferName>"
```

`<fillType>`	Fill the buffer continuously: `CONTinuous` Fill the buffer, then stop: `ONCE`
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

When a reading buffer is set to fill once, no data is overwritten in the buffer. When the buffer is filled, no more data is stored in that buffer and new readings are discarded.

When a reading buffer is set to fill continuously, the oldest data is overwritten by the newest data after the buffer fills.

When you change the fill mode of a buffer, any data in the buffer is cleared.

Example

`TRACe:MAKE "testData", 100`
 	
`TRACe:FILL:MODE? "testData"`
`TRACe:FILL:MODE CONT, "testData"`
`TRACe:FILL:MODE? "testData"`
 	
`TRACe:FILL:MODE?`
	Create a user-defined reading buffer named `testData` with a capacity of 100 measurements.
	Query the fill mode setting for `testData`.
	Output:
	`ONCE`
	Set `testData` fill mode to continuous.
	Query the fill mode setting for `testData`.
	Output:
	`CONT`
	Query the fill mode setting for `defbuffer1`.
	Output:
	`CONT`

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
:TRACe:CLEar (on page 6-146)

:TRACe:LOG:STATe

This command indicates if information events are logged when the specified reading buffer is at 0 % or 100 % filled.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	defbuffer1: 1 (ON) defbuffer2: 1 (ON) User-created buffer: 0 (OFF)

Usage

```
:TRACe:LOG:STATe <logState>
:TRACe:LOG:STATe <logState>, "<bufferName>"
:TRACe:LOG:STATe?
:TRACe:LOG:STATe? "<bufferName>"
```

`<logState>`	Do not log information events: `OFF` or `0` Log information events: `ON` or `1`
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

If this is set to on, when the reading buffer is cleared (0 % filled) or full (100 % filled), an event is logged in the event log. If this is set to off, reading buffer status is not reported in the event log.

Example

`TRACe:LOG:STATe?`	Query the log state of `defbuffer1`. Output: `1` Indicates that the log state is on.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
Using the event log (on page 2-152)

:TRACe:MAKE

This command creates a user-defined reading buffer.

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

Usage

```
:TRACe:MAKE "<bufferName>", <bufferSize>
:TRACe:MAKE "<bufferName>", <bufferSize>, <bufferStyle>
```

`<bufferName>`	A user-supplied string that indicates the name of the buffer
`<bufferSize>`	A number that indicates the maximum number of readings that can be stored in `<bufferName>`; minimum is 10
`<bufferStyle>`	The type of reading buffer to create: • Store readings with reduced accuracy (6.5 digits) with no formatting information, 1 μs accurate timestamp, maximum 27,500,000 readings: `COMPact` • Store readings with full accuracy with formatting, maximum 11,000,000 readings: `STANdard` (default) • Store the same information as standard, plus additional information, such as the ratio component of a DCV ratio measurement: `FULL` • Store external reading buffer data: `WRITable`

Details

You cannot assign user-defined reading buffers the name `defbuffer1` or `defbuffer2`.

If you create a reading buffer that has the same name as an existing user-defined buffer, the existing buffer is overwritten by the new buffer. Any data in the existing buffer is lost.

The new buffer becomes the active reading buffer.

The default fill mode of a user-defined buffer is once. You can change it to continuous.

The buffer style cannot be changed.

Writable readings are used to bring external data into the instrument. You cannot assign them to collect data from the instrument.

You can change the buffer capacity for an existing buffer through the front panel or by using the `:TRACe:POINts` command.

Example

`TRACe:MAKE "capTrace", 200`	This example creates a 200-element reading buffer named `capTrace`.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:FILL:MODE (on page 6-150)
:TRACe:POINts (on page 6-155)

:TRACe:WRITe:FORMat

This command sets the units and number of digits of the readings that are written into the reading buffer.

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

Usage

```
:TRACe:WRITe:FORMat "<bufferName>", <units>, <displayDigits>
```

`<bufferName>`	A user-supplied string that indicates the name of the buffer
`<units>`	The units for the measurement: • AMP • AMP_AC • CELSius • DECibel • FAHRenheit • FARad • HERTz • KELVin • NONE • OHM • PERCent • RATio • RECiprocal • SECond • VOLT • VOLT_AC • WATT • X
`<displayDigits>`	Integer from 3 to 7

Details

Defines the units and the number of digits that are reported for the data. It only affects how the data is shown in the reading buffer.

Example

`:TRAC:MAKE "write2me", 1000, WRITABLE`
`:TRAC:WRIT:FORM "write2me", WATT, 4`
`:TRAC:WRIT:READ "write2me", 1`
`:TRAC:WRIT:READ "write2me", 2`
`:TRAC:WRIT:READ "write2me", 3`
`:TRAC:WRIT:READ "write2me", 4`
`:TRAC:WRIT:READ "write2me", 5`
`    :TRAC:WRIT:READ "write2me", 6`
`:TRAC:DATA? 1, 6, "write2me", read,`
`    unit`
	Creates a 1000-point reading buffer named `write2me`. Style is writable.
	Set the data format to show units of watts with 4-½ digit resolution.
	Write 6 pieces of data into the buffer.
	Read the buffer.
	Output:
	`1.000000E+00,Watt DC,2.000000E+00,Watt`
	`    DC,3.000000E+00,Watt`
	`    DC,4.000000E+00,Watt`
	`    DC,5.000000E+00,Watt`
	`    DC,6.000000E+00,Watt DC`

Also see

Reading buffers (on page 3-12)

:TRACe:WRITe:READing

This command allows you to write readings into the reading buffer.

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

Usage

```
:TRACe:WRITe:READing "<bufferName>", <readingValue>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <seconds>
:TRACe:WRITe:READing "<bufferName>", <readingValue>, <seconds>, <fractionalSeconds>
```

`<bufferName>`	A user-supplied string that indicates the name of the buffer
`<readingValue>`	The value that is recorded in the buffer
`<seconds>`	An integer that represents the seconds
`<fractionalSeconds>`	An integer that represents the fractional seconds

Details

This command writes the data you specify into a reading buffer. The reading buffer must be set to the writable style, which is set when you make the buffer.

Data must be added in chronological order. If the time is not specified for a reading, it is set to one integer second after the last reading. As you write the data, the front-panel home screen updates and displays the reading you entered.

Example

```:TRAC:MAKE "write2me", 1000, WRITABLE```
```:TRAC:WRIT:FORM "write2me", WATT, 4```
```:TRAC:WRIT:READ "write2me", 1```
```:TRAC:WRIT:READ "write2me", 2```
```:TRAC:WRIT:READ "write2me", 3```
```:TRAC:WRIT:READ "write2me", 4```
```:TRAC:WRIT:READ "write2me", 5```
```   :TRAC:WRIT:READ "write2me", 6```
```:TRAC:DATA? 1, 6, "write2me", read,```
```   unit```
	Creates a 1000-point reading buffer named `write2me`. Style is writable.
	Set the data format to show a unit of watts with 4-½ digit resolution.
	Write 6 pieces of data into the buffer.
	Read the buffer.
	Output:
	```1.000000E+00,Watt DC,2.000000E+00,Watt```
	``` DC,3.000000E+00,Watt```
	``` DC,4.000000E+00,Watt```
	``` DC,5.000000E+00,Watt```
	``` DC,6.000000E+00,Watt DC```

Also see

Reading buffers (on page 3-12)
:TRACe:DATA? (on page 6-147)
:TRACe:WRITe:FORMat (on page 6-153)

:TRACe:POINts

This command contains the number of readings a buffer can store.

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

Usage

```
:TRACe:POINts <newSize>
:TRACe:POINts <newSize>, "<bufferName>"
:TRACe:POINts?
:TRACe:POINts? "<bufferName>"
```

`<newSize>`	The new size for the buffer
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This command allows you to change or view how many readings a buffer can store.

Example

```TRACe:MAKE "testData", 100```
```TRACe:POINts 300, "testData"```
```TRACe:POINts? "testData"```
 	
```TRACe:POINts?```
	Create a user-defined reading buffer named `testData` with a capacity of 100 measurements.
	Change the buffer capacity to 300.
	Query the capacity of `testData`.
	Output:
	`300`
	Query the capacity of the default buffer.
	Output:
	`10000`

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)

:TRACe:SAVE

This command saves data from the specified reading buffer to a USB flash drive.

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

Usage

```
:TRACe:SAVE "<fileName>"
:TRACe:SAVE "<fileName>", "<bufferName>"
:TRACe:SAVE "<fileName>", "<bufferName>", <timeFormat>
:TRACe:SAVE "<fileName>", "<bufferName>", <timeFormat>, <start>, <end>
```

`<fileName>`	A string that indicates the name of the file on the USB flash drive in which to save the reading buffer
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<timeFormat>`	Defines how date and time information from the buffer is saved in the file on the USB flash drive; the values are: • Dates, times, and fractional seconds are saved; the default value: `FORMat` • Relative timestamps are saved: `RELative` • Seconds and fractional seconds are saved: `RAW` • Timestamps are saved: `STAMp`
`<start>`	Defines the starting point in the buffer to start saving data
`<end>`	Defines the ending point in the buffer to stop saving data

Details

The filename must specify the full path (including `/usb1/`). If included, the file extension must be set to `.csv` (if no file extension is specified, `.csv` is added).

For options that save more than one item of time information, each item is comma-delimited. For example, the default format is `date`, `time`, and `fractional seconds` for each reading.

The Model DMM7510 does not check for existing files when you save. Verify that you are using a unique name to avoid overwriting any existing `.csv` files on the flash drive.

Example

| ```
TRACe:MAKE "MyBuffer", 100
SENSe:COUNt 5
MEASure:CURRent:DC? "MyBuffer"
TRACe:DATA? 1,5, "MyBuffer", READ, REL
TRACe:SAVE "/usb1/myData.csv", "MyBuffer"
TRACe:SAVE "/usb1/myDataRel.csv", "MyBuffer", REL
``` | Create a buffer called `MyBuffer` with a maximum size of 100. Make five readings for each measurement request and return the data. Make the measurements. Read the reading and relative timestamp value for each point  from 1 to 5. Output: `-0.000000,0.000000,` `-0.000000,` `0.301759,-0.000000,0.579068,-0.000000,` `0.884302,-0.000000,1.157444`Save all reading and default time information from a buffer named `MyBuffer` to a file named `myData.csv` on the USB flash drive. Save all readings and relative timestamps from `MyBuffer` to a file named `myDataRel.csv` on the USB flash drive. |

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
:TRACe:SAVE:APPend (on page 6-158)

:TRACe:SAVE:APPend

This command appends data from the reading buffer to a file on the USB flash drive.

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

Usage

```
:TRACe:SAVE:APPend "<fileName>"
:TRACe:SAVE:APPend "<fileName>", "<bufferName>"
:TRACe:SAVE:APPend "<fileName>", "<bufferName>", <timeFormat>
:TRACe:SAVE:APPend "<fileName>", "<bufferName>", <timeFormat>, <start>, <end>
```

`<fileName>`	A string that indicates the name of the file on the USB flash drive in which to save the reading buffer
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<timeFormat>`	Indicates how date and time information from the buffer is saved in the file on the USB flash drive; the values are: • Dates, times, and fractional seconds are saved; the default value: `FORMat` • Relative timestamps are saved: `RELative` • Seconds and fractional seconds are saved: `RAW` • Timestamps are saved: `STAMp`
`<start>`	Defines the starting point in the buffer to start saving data
`<end>`	Defines the ending point in the buffer to stop saving data

Details

If the file you specify does not exist on the USB flash drive, this command creates the file.

For options that save more than one item of time information, each item is comma-delimited. For example, the default format is date, time, and fractional seconds for each reading.

The file extension `.csv` is appended to the filename if necessary. Any file extension other than `.csv` generates an error.

The index column entry in the `.csv` file starts at 1 for each append operation.

Example

```
TRACe:MAKE "testData", 100
SENSe:COUNt 5
MEASure:CURRent:DC? "testData", READ, REL
TRACe:SAVE "/usb1/myData5.csv", "testData"
TRACe:CLEAr
MEASure:CURRent:DC?
TRACe:SAVE:APPend "/usb1/myData5.csv", "defbuffer1"
MEASure:CURRent:DC? "testData"
TRACe:SAVE:APPend "/usb1/myData5.csv", "testData", RAW, 6, 10
```

Create a buffer called testData.

Make 5 readings and return the fifth point, which will contain the reading and relative timestamp value. Store the buffer data in the myData5.csv file.

Clear defbuffer1.

Make 5 readings, store them in defbuffer1, and return the fifth reading.

Append all the readings stored in defbuffer1 to the myData5.csv file.

Take 5 more readings, store them in testData, and return the fifth reading.

Append all the readings stored in positions 6 through 10 testData to the myData5.csv file using raw timestamps.

Also see

:TRACe:STATistics:AVERage?

This command returns the average of all readings in the buffer.

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

Usage

```
:TRACe:STATistics:AVERage?
:TRACe:STATistics:AVERage? "<bufferName>"
```

<bufferName>	A string that indicates the reading buffer; the default buffers (defbuffer1 or defbuffer2) or the name of a user-defined buffer; if no buffer is specified, defbuffer1 is used

Details

This command returns the average reading calculated from all of the readings in the specified reading buffer.

Example

`TRACe:STAT:AVERage?`	Returns the average reading for the readings in the default buffer `defbuffer1`.
`TRACe:STAT:AVERage? "testData"`	Returns the average reading for the readings in the user-defined buffer `testData`.

Also see

:TRACe:STATistics:CLEar

This command clears the statistical information associated with the specified buffer.

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

Usage

```
:TRACe:STATistics:CLEar
:TRACe:STATistics:CLEar "<bufferName>"
```

`<bufferName>`	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; if no buffer is defined, clears the statistics from `defbuffer1`

Details

This command clears the statistics without clearing the readings.

Example

`TRACe:STATistics:CLEar`	Clear all statistics in `defbuffer1`.
`TRACe:STATistics:CLEar "testData"`	Clears all statistics in a user-defined buffer named `testData`.

Also see

:TRACe:STATistics:MAXimum?

This command returns the maximum reading value in the reading buffer.

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

Usage

```
:TRACe:STATistics:MAXimum?
:TRACe:STATistics:MAXimum? "<bufferName>"
```

<bufferName>	A string that indicates the reading buffer; the default buffers (defbuffer1 or defbuffer2) or the name of a user-defined buffer; if no buffer is specified, defbuffer1 is used

Example

TRACe:STAT:MAXimum?	Returns the maximum reading value in the default buffer, defbuffer1.
TRACe:STAT:MAXimum? "testData"	Returns the maximum reading value in the user-defined buffer testData.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
:TRACe:STATistics:AVERage? (on page 6-159)
:TRACe:STATistics:CLEar (on page 6-160)
:TRACe:STATistics:MINimum? (on page 6-162)
:TRACe:STATistics:PK2Pk? (on page 6-163)
:TRACe:STATistics:STDDev? (on page 6-163)

:TRACe:STATistics:MINimum?

This command returns the minimum reading value in the reading buffer.

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

Usage

```
:TRACe:STATistics:MINimum?
:TRACe:STATistics:MINimum? "<bufferName>"
```

<bufferName>	A string that indicates the reading buffer; the default buffers (defbuffer1 or defbuffer2) or the name of a user-defined buffer; if no buffer is specified, defbuffer1 is used

Example

TRACe:STAT:MINimum?	Returns the minimum reading value in the default buffer defbuffer1.
TRACe:STAT:MINimum? "testData"	Returns the minimum reading value in the user-defined buffer testData.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
:TRACe:STATistics:AVERage? (on page 6-159)
:TRACe:STATistics:CLEar (on page 6-160)
:TRACe:STATistics:MAXimum? (on page 6-161)
:TRACe:STATistics:PK2Pk? (on page 6-163)
:TRACe:STATistics:STDDev? (on page 6-163)

:TRACe:STATistics:PK2Pk?

This command returns the peak-to-peak value of all readings in the reading buffer.

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

Usage

```
:TRACe:STATistics:PK2Pk?
:TRACe:STATistics:PK2Pk? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:STAT:PK2Pk?`	Returns the peak-to-peak reading value in the default buffer `defbuffer1`.
`TRACe:STAT:PK2Pk? "testData"`	Returns the peak-to-peak reading value in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
:TRACe:STATistics:AVERage? (on page 6-159)
:TRACe:STATistics:CLEar (on page 6-160)
:TRACe:STATistics:MAXimum? (on page 6-161)
:TRACe:STATistics:MINimum? (on page 6-162)
:TRACe:STATistics:STDDev? (on page 6-163)

:TRACe:STATistics:STDDev?

This command returns the standard deviation of all readings in the buffer.

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

Usage

```
:TRACe:STATistics:STDDev?
:TRACe:STATistics:STDDev? "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Example

`TRACe:STAT:STDDev?`	Returns the standard deviation of the readings in the default buffer `defbuffer1`.
`TRACe:STAT:STDDev? "testData"`	Returns the standard deviation of the readings in the user-defined buffer `testData`.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
:TRACe:MAKE (on page 6-152)
:TRACe:STATistics:CLEar (on page 6-160)
:TRACe:STATistics:MAXimum? (on page 6-161)
:TRACe:STATistics:MINimum? (on page 6-162)
:TRACe:STATistics:PK2Pk? (on page 6-163)

:TRACe:TRIGger

This command makes readings using the active measure function and stores them in a reading buffer.

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

Usage

```
:TRACe:TRIGger
:TRACe:TRIGger "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

A measure function must be selected before sending this command.
This command makes the number of measurements that is set by the count command.

Example

`TRACe:MAKE "MyBuffer", 100`
`COUN 5`
`TRACe:TRIG "MyBuffer"`
`TRACe:DATA? 1,5, "MyBuffer", rel`
	Create a buffer called `MyBuffer` with a maximum size of 100.
	Make readings and store them in `MyBuffer`.
	Recall the relative time when the data points were measured for the first five readings in the buffer.
	Example output:
	`0.000000,0.408402,0.816757,1.208823,1.617529`

Also see

[:SENSe[1]]:COUNt (on page 6-116)
[:SENSe[1]]:FUNCtion[:ON] (on page 6-118)
:TRACe:MAKE (on page 6-152)

:TRACe:TRIGger:DIGitize

This command makes readings using the active digitize function and stores them in the reading buffer.

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

Usage

```
:TRACe:TRIGger:DIGitize
:TRACe:TRIGger:DIGitize "<bufferName>"
```

`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

A digitize function must be selected before sending this command.

This command makes the number of digitize measurements that is set by the digitize count command.

Example

```DIG:FUNC "VOLTage"```
```TRACe:MAKE "MyBuffer", 100```
```TRACe:TRIG:DIG "MyBuffer"```
```TRACe:TRIG:DIG "MyBuffer"```
```TRACe:TRIG:DIG "MyBuffer"```
```TRACe:TRIG:DIG "MyBuffer"```
```TRACe:TRIG:DIG "MyBuffer"```
```TRACe:DATA? 1,5, "MyBuffer", rel```
	Make the digitize voltage measurement function the active function.
	Create a buffer called `MyBuffer` with a maximum size of 100.
	Make readings and store them in `MyBuffer`.
	Recall the relative time when the data points were measured for the first five readings in the buffer.
	Example output:
	```0.000000,0.408402,0.816757,1.208823,1.617529```

Also see

[:SENSe[1]]:DIGitize:COUNt (on page 6-117)
[:SENSe[1]]:DIGitize:FUNCtion[:ON] (on page 6-118)
:TRACe:MAKE (on page 6-152)

TRIGger subsystem

The commands in this subsystem configure and control the trigger operations, including the trigger model.

:ABORt

This command stops all trigger model commands on the instrument.

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

Usage

```
:ABORt
```

Details

When this command is received, the instrument stops the trigger model.

Also see

Aborting the trigger model (on page 3-95)
Trigger model (on page 3-72)

:INITiate[:IMMediate]

This command starts the trigger model.

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

Usage

```
:INITiate[:IMMediate]
```

Also see

Trigger model (on page 3-72)

:TRIGger:BLENder<n>:CLEar

This command clears the blender event detector and resets the overrun indicator of blender <n>.

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

Usage

> :TRIGger:BLENder<n>:CLEar
>
> | <n> | The blender number (1 or 2) |
> |-----|------------------------------|

Details

> This command sets the blender event detector to the undetected state and resets the overrun indicator of the event detector.

Example

> | :TRIG:BLEN2:CLE | Clears the event detector for blender 2. |
> |-----------------|--|

Also see

> None

:TRIGger:BLENder<n>:MODE

This command selects whether the blender performs OR operations or AND operations.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Trigger blender clear	Save settings	AND

Usage

> :TRIGger:BLENder<n>:MODE <operation>
> :TRIGger:BLENder<n>:MODE?
>
> | <n> | The blender number (1 or 2) |
> |-----|------------------------------|
> | <operation> | The type of operation:
• OR
• AND |

Details

This command selects whether the blender waits for any one event (`OR`) or waits for all selected events (`AND`) before signaling an output event.

Example 1

`:DIG:LINE3:MODE TRIG, IN`
`:DIG:LINE5:MODE TRIG, IN`
`:TRIG:BLEN1:MODE OR`
`:TRIG:BLEN1:STIM1 DIG3`
`:TRIG:BLEN1:STIM2 DIG5`
	Set digital I/O lines 3 and 5 as trigger in lines. Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.

Also see

:TRIGger:BLENder<n>:STIMulus<m> (on page 6-169)

:TRIGger:BLENder<n>:OVERrun?

This command indicates whether or not an event was ignored because of the event detector state.

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

Usage

`:TRIGger:BLENder<n>:OVERrun?`

`<n>`	The blender number (1 or 2)

Details

Indicates if an event was ignored because the event detector was already in the detected state when the event occurred. This is an indication of the state of the event detector that is built into the event blender itself.

This command does not indicate if an overrun occurred in any other part of the trigger model or in any other trigger object that is monitoring the event. It also is not an indication of an action overrun.

Example

`:TRIG:BLEN1:OVER?`	If an event was ignored, the output is `1`. If an event was not ignored, the output is `0`.

Also see

:TRIGger:BLENder<n>:CLEar (on page 6-167)

:TRIGger:BLENder<n>:STIMulus<m>

This command specifies the events that trigger the blender.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle Trigger blender clear	Save settings	NONE

Usage

```
:TRIGger:BLENder<n>:STIMulus<m> <event>
:TRIGger:BLENder<n>:STIMulus<m>?
```

`<n>`	The blender number (1 or 2)
`<m>`	The stimulus input number (1 to 4)
`<event>`	See **Details**

Details

There are four stimulus inputs that can each select a different event.
Use zero to disable the blender input.

The `<event>` parameter may be any of the trigger events shown in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

`:DIG:LINE3:MODE TRIG, IN`
`:DIG:LINE5:MODE TRIG, IN`
`:TRIG:BLEN1:MODE OR`
`:TRIG:BLEN1:STIM1 DIG3`
`:TRIG:BLEN1:STIM2 DIG5`
	Set digital I/O lines 3 and 5 as trigger in lines. Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.

Also see

:TRIGger:BLENder<n>:MODE (on page 6-167)

:TRIGger:BLOCk:BRANch:ALWays

This command defines a trigger model block that always goes to a specific block.

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

Usage

`:TRIGger:BLOCk:BRANch:ALWays <blockNumber>, <branchToBlock>`

`<blockNumber>`	The sequence of the block in the trigger model
`<branchToBlock>`	The block number of the trigger model block to execute when the trigger model reaches this block

Details

When the trigger model reaches a branch-always building block, it goes to the building block set by `<branchToBlock>`.

Example

`TRIG:BLOC:BRAN:ALW 9, 20`	When the trigger model reaches block 9, it will always branch to block 20.

Also see

None

:TRIGger:BLOCk:BRANch:COUNter

This command defines a trigger model block that branches to a specified block a specified number of times.

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

Usage

:TRIGger:BLOCk:BRANch:COUNter <blockNumber>, <targetCount>, <branchToBlock>

<blockNumber>	The sequence of the block in the trigger model
<targetCount>	The number of times to repeat
<branchToBlock>	The block number of the trigger model block to execute when the counter is less than to the <targetCount> value

Details

This command defines a trigger model building block that branches to another block using a counter to iterate a specified number of times.

Counters increment every time the trigger model reaches them until they are more than or equal to the count value. At that point, the trigger model continues to the next building block in the sequence.

If you are using remote commands, you can query the counter. The counter is incremented immediately before the branch compares the actual counter value to the set counter value. Therefore, the counter is at 0 until the first comparison. When the trigger model reaches the set counter value, branching stops and the counter value is one greater than the setting. Use :TRIGger:BLOCk:BRANch:COUNter:COUNt? to query the counter.

Example

| TRIG:LOAD "EMPTY"
TRIG:BLOC:BUFF:CLEAR 1
TRIG:BLOC:MEAS 2
TRIG:BLOC:BRAN:COUN 3, 5, 2
TRIG:BLOC:DEL:CONS 4, 1
TRIG:BLOC:BRAN:COUN 5, 3, 2 | Reset trigger model settings.
Clear defbuffer1 at the beginning of the trigger model.
Loop and take 5 readings.
Delay a second.
Loop three more times back to block 2.
At end of execution, 15 readings are stored in defbuffer1. |

Also see

:TRIGger:BLOCk:BRANch:COUNter:COUNt? (on page 6-172)

:TRIGger:BLOCk:BRANch:COUNter:COUNt?

This command returns the count that the trigger model is on.

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

Usage

```
:TRIGger:BLOCk:BRANch:COUNter:COUNt? <blockNumber>
```

<blockNumber>	The sequence of the block in the trigger model

Details

The query returns the number of times the trigger model has looped. The counter is defined by `:TRIGger:BLOCk:BRANch:COUNter`.

Example

`TRIG:LOAD "EMPTY"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
`INIT`
`TRIG:BLOCK:BRAN:COUN:COUN?`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 readings.
	Delay 1 s.
	Loop three more times back to block 2.
	At end of execution, 15 readings are stored in `defbuffer1`.
	Check to see which count the trigger model has completed.

Also see

:TRIGger:BLOCk:BRANch:COUNter:RESet

This command creates a block in the trigger model that resets a branch counter to 0.

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

Usage

```
:TRIGger:BLOCk:BRANch:COUNter:RESet <blockNumber>, <counter>
```

<blockNumber>	The sequence of the block in the trigger model
<counter>	The block number of the counter that is to be reset

Details

When the trigger model reaches the Counter Reset block, it resets the count of the specified Branch on Counter block to zero.

Example

`TRIG:LOAD "EMPTY"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
`TRIG:BLOC:BRAN:COUN:RES 6, 3`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 readings.
	Delay a second.
	Loop three more times back to block 2.
	Reset block 3 to 0.

Also see

:TRIGger:BLOCk:BRANch:COUNter (on page 6-171)
:TRIGger:BLOCk:BRANch:COUNter:COUNt? (on page 6-172)

:TRIGger:BLOCk:BRANch:DELTa

This command defines a trigger model block that goes to a specified block if the difference of two measurements meets preset criteria.

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

Usage

```
:TRIGger:BLOCk:BRANch:DELTa <blockNumber>, <targetDifference>, <branchToBlock>
:TRIGger:BLOCk:BRANch:DELTa <blockNumber>, <targetDifference>, <branchToBlock>,
    <measureBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<targetDifference>`	The value against which the block compares the difference between the measurements
`<branchToBlock>`	The block number of the trigger model block to execute when the difference between the measurements is less than or equal to the `<targetDifference>`
`<measureBlock>`	The block number of the measure or digitize block that makes the measurements to be compared; if this is `0` or undefined, the trigger model uses the previous measure or digitize block

Details

This block calculates the difference between the last two measurements from a measure or digitize block. It subtracts the most recent measurement from the previous measurement.

The difference between the measurements is compared to the target difference. If the difference is less than the target difference, the trigger model goes to the specified branching block. If the difference is more than the target difference, the trigger model proceeds to the next block in the trigger block sequence.

If you do not define the measure or digitize block, it will compare measurements of a measure or digitize block that precedes the branch delta block. For example, if you have a measure block, a wait block, another measure block, another wait block, and then the branch delta block, the delta block compares the measurements from the second measure block. If a preceding measure or digitize block does not exist, an error occurs.

Example

`TRIG:BLOC:BRAN:DELT 5, 0.5, 7, 4`	Configure trigger block 5 to compare the differences between the measurements made in block 4. If the difference between them is less the 0.5, branch to block 7.

Also see

Delta building block (on page 3-84)

:TRIGger:BLOCk:BRANch:EVENt

This command branches to a specified block when a specified trigger event occurs.

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

Usage

`:TRIGger:BLOCk:BRANch:EVENt <blockNumber>, <event>, <branchToBlock>`

`<blockNumber>`	The sequence of the block in the trigger model
`<event>`	The event that must occur before the trigger model branches the specified block
`<branchToBlock>`	The block number of the trigger model block to execute when the specified event occurs

Details

The branch-on-event building block goes to a branching block after a specified trigger event occurs. If the trigger event has not yet occurred when the trigger model reaches the branch-on-event block, the trigger model continues to execute the blocks in the normal sequence. After the trigger event occurs, the next time the trigger model reaches the branch-on-event block, it goes to the branching block.

If you set the branch event to none, an error is generated when you run the trigger model.

The following table shows the constants for the events.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

```
:TRIG:BLOC:BRAN:EVEN 6, DISP, 2
```
When the trigger model reaches this block, if the front-panel TRIGGER key has been pressed, the trigger model returns to block 2. If the TRIGGER key has not been pressed, the trigger model continues to block 7 (the next block in the trigger model).

Also see

On event building block (on page 3-85)

:TRIGger:BLOCk:BRANch:LIMit:CONStant

This command defines a trigger model block that branches to a block outside the normal trigger model flow if a measurement meets preset criteria.

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

Usage

```
:TRIGger:BLOCk:BRANch:LIMit:CONStant <blockNumber>, <limitType>, <limitA>,
   <limitB>, <branchToBlock>
:TRIGger:BLOCk:BRANch:LIMit:CONStant <blockNumber>, <limitType>, <limitA>,
   <limitB>, <branchToBlock>, <measureBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<limitType>`	The type of limit (ABOVe, BELow, INside, or OUTside)
`<limitA>`	The limit that the measurement is tested against; if *limitType* is set to: • ABOVe: This value is ignored • BELow: The measurement must be below this value • INside: The low limit that the measurement is compared against • OUTside: The low limit that the measurement is compared against
`<limitB>`	The upper limit that the measurement is tested against; if *limitType* is set to: • ABOVe: The measurement must be above this value • BELow: This value is ignored • INside: The high limit that the measurement is compared against • OUTside: The high limit that the measurement is compared against
`<branchToBlock>`	The block number of the trigger model block to execute when the measurement meets the defined criteria
`<measureBlock>`	The block number of the measure or digitize block that makes the measurements to be compared; if this is 0 or undefined, the trigger model uses the previous measure or digitize block

Details

The branch-on-constant-limits block goes to a branching block if a measurement meets the criteria set by this command.

The type of limit can be:

- Above: The measurement is above the value set by limit B; limit A must be set, but is ignored when this type is selected
- Below: The measurement is below the value set by limit A; limit B must be set, but is ignored when this type is selected
- Inside: The measurement is inside the values set by limits A and B; limit A must be the low value and Limit B must be the high value
- Outside: The measurement is outside the values set by limits A and B; limit A must be the low value and Limit B must be the high value

The measurement block must be a measure or digitize building block that occurs in the trigger model before the branch-on-constant-limits block. The last measurement from a measure or digitize building block is used.

If the limit A is more than the limit B, the values are automatically swapped so that the lesser value is used as the lower limit.

Example

```
TRIGger:BLOCk:BRANch:LIMit:CONStant 5, OUTside, 0.15, 0.65, 8
```
Configure trigger block 5 to check for measurements in the last measure or digitize block. If the measurements are outside of the 0.15 and 0.65 limits, branch to block 8.

Also see

[Constant Limit building block](#) (on page 3-81)

:TRIGger:BLOCk:BRANch:LIMit:DYNamic

This command defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

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

Usage

```
:TRIGger:BLOCk:BRANch:LIMit:DYNamic <blockNumber>, <limitType>, <limitNumber>,
    <branchToBlock>
:TRIGger:BLOCk:BRANch:LIMit:DYNamic <blockNumber>, <limitType>, <limitNumber>,
    <branchToBlock>, <measureBlock>
```

<blockNumber>	The sequence of the block in the trigger model
<limitType>	The type of limit (ABOVe, BELow, INside, or OUTside)
<limitNumber>	The limit number (1 or 2)
<branchToBlock>	The block number of the trigger model block to execute when the limits are met
<measureBlock>	The block number of the measure or digitize block that makes the measurements to be compared; if this is 0 or undefined, the trigger model uses the previous measure or digitize block

Details

The branch-on-dynamic-limits building block defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

When you define this building block, you set:

- The type of limit (above, below, inside, or outside the limit values)
- The limit number (you can have 1 or 2 limits)
- The building block to go to if the measurement meets the criteria
- The building block that makes the measurement that is compared to the limits; the last measurement from that building block is used

There are two user-defined limits: limit 1 and limit 2. Both include their own high and low values, which are set using the front-panel Calculations limit settings or through commands. The results of these limit tests are recorded in the reading buffer that accompanies each stored reading.

Limit values are stored in the measure configuration list, so you can use a configuration list to step through different limit values.

The measure or digitize block must occur in the trigger model before the branch-on-dynamic-limits block. If no measure or digitize block is defined, the measurement from the previous measure or digitize block is used. If no previous measure or digitize block exists, an error is reported.

Example

`CALC2:LIM1:STAT ON`
`CALC2:LIM1:LOW –5.17`
`CALC2:LIM1:UPP –4.23`
`TRIG:BLOC:BRAN:LIM:DYN 9, IN, 1, 12, 7`
	Set the limits on with a low limit of -5.17 and a high limit of -4.23. Set trigger block 9 to test if the limit is inside those limits based on the measurement reading at block 7. If the measurement is within the limits, go to block 12.

Also see

Dynamic Limits building block (on page 3-83)
:CALCulate2:<function>:LIMit<Y>:LOWer[:DATA] (on page 6-25)
:CALCulate2:<function>:LIMit<Y>:UPPer[:DATA] (on page 6-27)

:TRIGger:BLOCk:BRANch:ONCE

This command causes the trigger model to branch to a specified building block the first time it is encountered in the trigger model.

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

Usage

`:TRIGger:BLOCk:BRANch:ONCE <blockNumber>, <branchToBlock>`

`<blockNumber>`	The sequence of the block in the trigger model
`<branchToBlock>`	The block number of the trigger model block to execute when the trigger model first encounters this block

Details

The branch-once building block branches to a specified block the first time trigger model execution encounters the branch-once block. If it is encountered again, the trigger model ignores the block and continues in the normal sequence.

The once block is reset when trigger model execution reaches the idle state. Therefore, the branch-once block always executes the first time the trigger model execution encounters this block.

Example

```
:TRIG:BLOC:BRAN:ONCE 2, 4
```

The first time the trigger model reaches block 2, the trigger model goes to block 4 instead of proceeding to the default sequence of block 3.

Also see

Once building block (on page 3-83)
:TRIGger:BLOCk:BRANch:ONCE:EXCLuded (on page 6-179)

:TRIGger:BLOCk:BRANch:ONCE:EXCLuded

This command causes the trigger model to go to a specified building block every time the trigger model encounters it, except for the first time.

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

Usage

```
:TRIGger:BLOCk:BRANch:ONCE:EXCLuded <blockNumber>, <branchToBlock>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<branchToBlock>`	The block number of the trigger model block to execute when the trigger model encounters this block after the first encounter

Details

The branch-once-excluded building block is ignored the first time the trigger model encounters it. After the first encounter, the trigger model goes to the specified branching block.

The branch-once-excluded block is reset when the trigger model starts or is placed in idle.

Example

```
:TRIG:BLOC:BRAN:ONCE:EXCL 2, 4
```

When the trigger model reaches block 2 the first time, the trigger model goes to block 3. If the trigger model reaches this block again, the trigger model goes to block 4.

Also see

Once excluded building block (on page 3-84)
:TRIGger:BLOCk:BRANch:ONCE (on page 6-178)

:TRIGger:BLOCk:BUFFer:CLEar

This command defines a trigger model block that clears the reading buffer.

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

Usage

```
:TRIGger:BLOCk:BUFFer:CLEar <blockNumber>
:TRIGger:BLOCk:BUFFer:CLEar <blockNumber>, "<bufferName>"
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bufferName>`	The name of the buffer, which must be an existing buffer; if no buffer is defined, `defbuffer1` is used

Details

When trigger model execution reaches the buffer clear trigger block, the instrument empties the specified reading buffer. The specified buffer can be the default buffer or a buffer that you defined. If you are clearing a user-defined reading buffer, you must create the buffer before you define this block.

Example

``` TRIG:LOAD "EMPTY" TRIG:BLOC:BUFF:CLE 1 TRIG:BLOC:MEAS 2 TRIG:BLOC:BRAN:COUN 3, 5, 2 TRIG:BLOC:DEL:CONS 4, 1 TRIG:BLOC:BRAN:COUN 5, 3, 2 ```	Reset trigger model settings. Clear `defbuffer1` at the beginning of the trigger model. Loop and take 5 readings. Delay 1 s. Loop three more times back to block 2. At end of execution, 15 readings are stored in `defbuffer1`.

Also see

Buffer clear building block (on page 3-77)
:TRACe:MAKE (on page 6-152)

:TRIGger:BLOCk:CONFig:NEXT

This command recalls the settings at the next index point of a configuration list.

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

Usage

```
:TRIGger:BLOCk:CONFig:NEXT <blockNumber>, "<configurationList>"
```

`<blockNumber>`	The sequence of the block in the trigger model
`<configurationList>`	The configuration list from which to recall settings

Details

When trigger model execution reaches a configuration recall next building block, the settings at the next index point in the specified configuration list are restored.

The first time the trigger model encounters this block for a specific configuration list, the first index point is recalled. Each subsequent time this block is encountered, the settings at the next index point in the configuration list are recalled and take effect before the next step executes. When the last index point in the list is reached, it returns to the first point.

The configuration list must be defined before you can use this building block.

Example

`TRIG:BLOC:CONF:NEXT 12, "SETTINGS_LIST"`	Set trigger block 12 to restore the settings from the next index point that is stored in the configuration list `SETTINGS_LIST`.

Also see

[Configuration lists](#) (on page 3-32)

:TRIGger:BLOCk:CONFig:PREVious

This command defines a trigger model block that recalls the settings stored at the previous index point in a configuration list.

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

Usage

`:TRIGger:BLOCk:CONFig:PREVious <blockNumber>, "<configurationList>"`

`<blockNumber>`	The sequence of the block in the trigger model
`<configurationList>`	The configuration list from which to recall settings

Details

The Config List Prev building block defines a trigger model block that recalls the settings stored at the previous index point in a configuration list.

The configuration list previous index trigger block type recalls the previous index point in a configuration list. It configures the settings of the instrument based on the settings at that index. The trigger model executes the settings at that index before the next block is executed.

The first time the trigger model encounters this block, the last index in the configuration list is recalled. Each subsequent time trigger model execution reaches a configuration list previous block for this configuration list, it goes backward one index. When the first index in the list is reached, it goes to the last index in the configuration list.

You must create the configuration list before you can define it in this building block.

Example

`TRIG:BLOC:CONF:PREV 14, "SETTINGS_LIST"`	Set trigger block 14 to restore the settings from the previous index point that is stored in the configuration list `SETTINGS_LIST`.

Also see

[Configuration lists](#) (on page 3-32)

:TRIGger:BLOCk:CONFig:RECall

This command recalls the system settings that are stored in a configuration list.

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

Usage

```
:TRIGger:BLOCk:CONFig:RECall <blockNumber>, "<configurationList>"
:TRIGger:BLOCk:CONFig:RECall <blockNumber>, "<configurationList>", <point>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<configurationList>`	A string that defines the configuration list to recall
`<point>`	The point in the configuration list to recall; defaults to 1 if not selected

Details

When the trigger model reaches a configuration recall building block, the settings in the specified configuration list are recalled.
You can restore a specific set of configuration settings in the configuration list by defining the index.

Example

`TRIG:BLOCK:CONF:RECALL 1, "SETTINGS_LIST", 1`	Recall the settings in index point 1 of the `SETTINGS_LIST` configuration list as block 1 of the trigger model.

Also see

[Configuration lists](#) (on page 3-32)

:TRIGger:BLOCk:DELay:CONStant

This command adds a constant delay to the trigger model.

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

Usage

```
:TRIGger:BLOCk:DELay:CONStant <blockNumber>, <time>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<time>`	The amount of time to delay (167 ns to 10,000 s, or 0 for no delay)

Details

When trigger model execution reaches a delay building block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made, and if any previously executed block started infinite measurements, they also continue to be made.

This delay waits for the set amount of delay time to elapse before proceeding to the next block in the trigger model.

If other delays have been set, this delay is in addition to the other delays.

Example

`TRIG:LOAD "EMPTY"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 readings.
	Delay a second.
	Loop three more times back to block 2.
	At end of execution, 15 readings are stored in `defbuffer1`.

Also see

None

:TRIGger:BLOCk:DELay:DYNamic

This command adds a delay to the execution of the trigger model.

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

Usage

```
:TRIGger:BLOCk:DELay:DYNamic <blockNumber>, MEASure<n>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<n>`	The number of the user delay; 1 to 5 set by`[:SENSe[1]]:<function>:DELay:USER<n>`

Details

When trigger model execution reaches a dynamic delay building block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made.

Each measure and digitize function can have up to 5 unique user delay times (M1 to M5). The delay time is set by the user-delay command. This command is only available over a remote interface.

Example

```FUNC "VOLT"```
```:VOLT:DEL:USER1 5```
```:TRIG:LOAD "EMPTY"```
```:TRIG:BLOC:BUFF:CLEAR 1```
```:TRIG:BLOC:MEAS 2```
```:TRIG:BLOC:BRAN:COUN 3, 5, 2```
```:TRIG:BLOC:DEL:DYN 4, MEAS1```
```:TRIG:BLOC:BRAN:COUN 5, 3, 2```
```:INIT```
	Set function to DC voltage.
	Set user delay 1 for DC voltage measurements to 5 s.
	Clear the trigger model.
	Set trigger block 1 to clear the reading buffer.
	Set trigger block 2 to make a measurement.
	Set trigger block 3 to loop and make 5 measurements.
	Set trigger block 4 to a dynamic delay, using user delay 1.
	Set trigger block 5 to loop 3 times.
	Start the trigger model.

Also see

[:SENSe[1]]:<function>:DELay:USER<n> (on page 6-77)

:TRIGger:BLOCk:DIGital:IO

This command defines a trigger model block that sets the lines on the digital I/O port high or low.

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

Usage

```
:TRIGger:BLOCk:DIGital:IO <blockNumber>, <bitPattern>
:TRIGger:BLOCk:DIGital:IO <blockNumber>, <bitPattern>, <bitMask>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bitPattern>`	Sets the value that specifies the output line bit pattern (0 to 63)
`<bitMask>`	Specifies the bit mask; if omitted, all lines are driven (0 to 63)

Details

To set the lines on the digital I/O port high or low, you can send a bit pattern that is specified as an integer value. The least significant bit maps to digital I/O line 1 and the most significant bit maps to digital I/O line 6.

The output line bit mask defines the bits in the pattern that are driven high or low. If the bit for a line is set to 1, the line is driven high. If the bit is set to 0, the line is driven low. A binary 1 in the bit mask indicates that the corresponding I/O line should be driven according to the bit pattern. To drive all lines, specify all ones (63) or omit this parameter.

For this building block to work as expected, make sure you configure the trigger type and line state of the digital line for use with the trigger model (use the digital line mode command).

Example

```:DIGital:LINE3:MODE DIG,OUT :DIGital:LINE4:MODE DIG,OUT :DIGital:LINE5:MODE DIG,OUT :DIGital:LINE6:MODE DIG,OUT :TRIG:BLOC:DIG:IO 4, 20, 60```	The first four lines of code configures digital I./O lines 3 through 6 as digital outputs. Trigger block 4 is then configured with a bit pattern of 20 (digital I/O lines 3 and 5 high). The optional bit mask is specified as 60 (lines 3 through 6), so both lines 3 and 5 are driven high.

Also see

:DIGital:LINE<n>:MODE (on page 6-34)
Digital I/O bit weighting (on page 3-53)
Digital I/O port configuration (on page 3-45)

:TRIGger:BLOCk:DIGitize

This command defines a trigger block that makes a measurement using a digitize function.

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

Usage

```
:TRIGger:BLOCk:DIGitize <blockNumber>
:TRIGger:BLOCk:DIGitize <blockNumber>, "<bufferName>"
:TRIGger:BLOCk:DIGitize <blockNumber>, "<bufferName>", <count>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bufferName>`	The name of the buffer, which must be an existing buffer; if no buffer is defined, `defbuffer1` is used
`<count>`	Specifies the number of readings to make before moving to the next block in the trigger model; set to a specific value or to infinite by setting to `INF`; to stop the count, set to `0`; if not specified, defaults to 1

Details

When trigger model execution reaches the block:

1. The instrument begins making a measurement.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurement into the specified reading buffer.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and measurements continue in the background until the trigger model execution reaches another digitize block or until the trigger model is aborted or re-initialized.

A digitize function (digitize voltage or digitize current) must be selected before you run a trigger model that contains this block. You cannot have a measure block and a digitize block in the same trigger model.

Example

```TRIG:LOAD "EMPTY"```
```TRIG:BLOC:BUFF:CLE 1```
```TRIG:BLOC:DIG 2```
```TRIG:BLOC:BRAN:COUN 3, 5, 2```
```TRIG:BLOC:DEL:CONS 4, 1```
```TRIG:BLOC:BRAN:COUN 5, 3, 2```
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of the trigger model.
	Loop and take 5 digitized readings.
	Delay 1 s.
	Loop three more times back to block 2.
	At end of execution, 15 readings are stored in `defbuffer1`.

Also see

Digitize building block (on page 3-76)
:TRACe:MAKE (on page 6-152)

:TRIGger:BLOCk:LIST?

This command returns the settings for all trigger model building blocks.

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

Usage

```
:TRIGger:BLOCk:LIST?
```

Details

This returns the settings for the trigger model.

Example

:TRIG:BLOC:LIST?	Returns the settings for the trigger model. Example output is:
	1) BUFFER_CLEAR BUFFER: defbuffer1
	2) MEASURE BUFFER: defbuffer1 COUNT: 1
	3) BRANCH_COUNTER VALUE: 5 BRANCH_BLOCK: 2
	4) DELAY_CONSTANT DELAY: 1.000000000
	5) BRANCH_COUNTER VALUE: 3 BRANCH_BLOCK: 2

Also see

None

:TRIGger:BLOCk:LOG:EVENt

This command allows you to log an event in the event log when the trigger model is running.

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

Usage

:TRIGger:BLOCk:LOG:EVENt <blockNumber>, <eventNumber>, "<message>"

`<blockNumber>`	The sequence of the block in the trigger model
`<eventNumber>`	The event number: • INFO<n> • WARNing<n> • ERRor<n> Where <n> is 1 to 4; you can define up to four of each type You can also set ABORt, which aborts the trigger model immediately and posts a warning event log message
`<message>`	A string up to 31 characters

Details

This block allows you to log an event in the event log when trigger model execution reaches this block. You can also force the trigger model to abort with this block. When the trigger model executes the block, the defined event is logged. If the abort option is selected, the trigger model is also aborted immediately.

You can define the type of event (information, warning, abort model, or error). All events generated by this block are logged in the event log. Warning and error events are also displayed in a popup on the front-panel display.

Note that using this block too often in a trigger model could overflow the event log. It may also take away from the time needed to process more critical trigger model blocks.

Example

TRIGger:BLOCk:LOG:EVENt 9, INFO2, "Trigger model complete"	Set trigger model block 9 to log an event when the trigger model completes.

Also see

None

:TRIGger:BLOCk:MEASure

This command defines a trigger block that makes a measurement.

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

Usage

```
:TRIGger:BLOCk:MEASure <blockNumber>
:TRIGger:BLOCk:MEASure <blockNumber>, "<bufferName>"
:TRIGger:BLOCk:MEASure <blockNumber>, "<bufferName>", <count>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<bufferName>`	The name of the buffer, which must be an existing buffer; if no buffer is defined, `defbuffer1` is used
`<count>`	Specifies the number of readings to make before moving to the next block in the trigger model; set to a specific value or to infinite by setting to `INF`; to stop the count, set to `0`; if not specified, defaults to 1

Details

When trigger model execution reaches the block:

1. The instrument begins making a measurement.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurement into the specified reading buffer.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and measurements continue in the background until the trigger model execution reaches another measure block or until the trigger model ends.

You must select a measure function before running a trigger model that contains this block.

You cannot include a measure block and a digitize block in the same trigger model.

Example

``` TRIG:LOAD "EMPTY" TRIG:BLOC:BUFF:CLEAR 1, "defbuffer2" TRIG:BLOC:MEAS 2, "defbuffer2" TRIG:BLOC:BRAN:COUN 3, 5, 2 TRIG:BLOC:DEL:CONS 4, 1 TRIG:BLOC:BRAN:COUN 5, 3, 2 ```	Reset trigger model settings. Clear `defbuffer2` at the beginning of the trigger model. Set the measurements to be stored in `defbuffer2`. Loop and take 5 readings. Delay 1 s. Loop three more times back to block 2. At end of execution, 15 readings are stored in `defbuffer2`.

Also see

Measure building block (on page 3-75)
:TRACe:MAKE (on page 6-152)

:TRIGger:BLOCk:NOP

This command creates a placeholder that performs no action in the trigger model.

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

Usage

```
:TRIGger:BLOCk:NOP <blockNumber>
```

<blockNumber>	The sequence of the block in the trigger model

Details

If you remove a trigger model block, you can use this block as a placeholder for the block number so that you do not need to renumber the other blocks.

Example

TRIG:BLOC:NOP 5	Set block number 5 to be a no operation block.

Also see

None

:TRIGger:BLOCk:NOTify

This command defines a trigger model block that generates a trigger event and immediately continues to the next block.

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

Usage

```
:TRIGger:BLOCk:NOTify <blockNumber>, <notifyID>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<notifyID>`	The identification number of the notification; 1 to 8

Details

When trigger model execution reaches a notify block, the instrument generates a trigger event and immediately continues to the next block.

Other commands can reference the event that the notify block generates. This assigns a stimulus somewhere else in the system. For example, you can use the notify event as the stimulus of a hardware trigger line, such as a digital I/O line.

When you call this event, you use the format `NOTIFY` followed by the notify identification number. For example, if you assign `<notifyID>` as 4, you would refer to it as `NOTIFY4` in the command that references it.

Example

`:TRIG:BLOC:NOT 5, 2` `:TRIG:BLOC:BRAN:EVEN 6, NOTIFY2, 2`	Define trigger model block 5 to be the notify 2 event. Assign the notify 2 event to be the trigger for stimulus for the branch event for block 6.

Also see

Notify building block (on page 3-80)

:TRIGger:BLOCk:WAIT

This command defines a trigger model block that waits for an event before allowing the trigger model to continue.

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

Usage

```
:TRIGger:BLOCk:WAIT <blockNumber>, <event>
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <logic>, <event>
:TRIGger:BLOCk:WAIT <blockNumber>, <event>, <logic>, <event>, <event>
```

`<blockNumber>`	The sequence of the block in the trigger model
`<event>`	The event that must occur before the trigger block allows trigger execution to continue; see **Details** for event names
`<logic>`	If each event must occur before the trigger model continues: `AND` If at least one of the events must occur before the trigger model continues: `OR`

Details

You can use the wait block to synchronize measurements with other instruments and devices.

You can set the instrument to wait for the following events:

- Front-panel TRIGGER key press
- Notify
- Command interface trigger
- Digital input/output signals, such as DB-9 and TSP-Link
- LAN
- Blender
- Timer
- Analog trigger
- External in trigger

The event can occur before trigger model execution reaches the wait block. If the event occurs after trigger model execution starts but before the trigger model execution reaches the wait block, the trigger model records the event. When trigger model execution reaches the wait block, it executes the wait block without waiting for the event to happen again.

The instrument clears the memory of the recorded event when trigger model execution is at the start block and when the trigger model exits the wait block.

All items in the list are subject to the same action; you cannot combine `AND` and `OR` logic in a single block.

You cannot leave the first event as no trigger. If the first event is not defined, the trigger model errors when you attempt to initiate it.

The following table shows the constants for the events.

Trigger events

Event description	Event constant
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

`:TRIGger:BLOCk:WAIT 9, DISP`	Set trigger model block 9 to wait for a user to press the TRIGGER key on the front panel before continuing.

Also see

:TRIGger:DIGital<n>:IN:CLEar

This command clears the trigger event on a digital input line.

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

Usage

```
:TRIGger:DIGital<n>:IN:CLEar
```

<n>	Digital I/O trigger line (1 to 6)

Details

The event detector of a trigger enters the detected state when an event is detected. For the specified trigger line, this command clears the event detector, discards the history, and clears the overrun status(sets the overrun status to `false`).

For this building block to work as expected, make sure you configure the trigger type and line state of the digital line for use with the trigger model (use the digital line mode command).

Example

:TRIG:DIG2:IN:CLE	Clears the trigger event detector on I/O line 2.

Also see

:DIGital:LINE<n>:MODE (on page 6-34)
Digital I/O port configuration (on page 3-45)
:TRIGger:DIGital<n>:IN:OVERrun? (on page 6-196)

:TRIGger:DIGital<n>:IN:EDGE

This command sets the edge used by the trigger event detector on the given trigger line.

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

Usage

```
:TRIGger:DIGital<n>:IN:EDGE <detectedEdge>
:TRIGger:DIGital<n>:IN:EDGE?
```

<n>	Digital I/O trigger line (1 to 6)
<detectedEdge>	The trigger edge value: • Detect falling-edge triggers as inputs: `FALLing` • Detect rising-edge triggers as inputs: `RISing` • Detect either falling or rising-edge triggers as inputs: `EITHer` See **Details** for descriptions of values

Details

The input state of the digital I/O line is controlled by the type of edge specified by this command.

To directly control the line state, set the mode of the line to digital and use the write command. When the digital line mode is set for open drain, the edge settings assert a TTL low-pulse.

Trigger mode values

Value	Description
FALLing	Detects falling-edge triggers as input when the line is configured as an input or open drain.
RISing	Detects rising-edge triggers as input when the line is configured as an open drain.
EITHer	Detects rising- or falling-edge triggers as input when the line is configured as an input or open drain.

Example

`:DIG:LINE4:MODE TRIG,IN` `:TRIG:DIG4:IN:EDGE RIS`	Sets the input trigger mode for the digital I/O line 4 to detect rising-edge triggers as input.

Also see

Digital I/O port configuration (on page 3-45)
:DIGital:LINE<n>:MODE (on page 6-34)
:DIGital:WRITe <n> (on page 6-38)
:TRIGger:DIGital<n>:IN:CLEar (on page 6-194)

:TRIGger:DIGital<n>:IN:OVERrun?

This command returns the event detector overrun status.

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

Usage

```
:TRIGger:DIGital<n>:IN:OVERrun?
```

<n>	Digital I/O trigger line (1 to 6)

Details

This command returns the event detector overrun status as `0` (false) or `1` (true).

If this is `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

TRIG:DIG1:IN:OVER?	Returns `0` if no overruns have occurred or `1` if one or more overrun have occurred for I/O line 1.

Also see

Digital I/O port configuration (on page 3-45)
:DIGital:LINE<n>:MODE (on page 6-34)

:TRIGger:DIGital<n>:OUT:LOGic

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

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

Usage

```
:TRIGger:DIGital<n>:OUT:LOGic <logicType>
:TRIGger:DIGital<n>:OUT:LOGic?
```

<n>	Digital I/O trigger line (1 to 6)
<logicType>	The output logic of the trigger generator: • Assert a TTL-high pulse for output: `POSitive` • Assert a TTL-low pulse for output: `NEGative`

Details

This command sets the trigger event generator to assert a TTL pulse for output logic. Positive is a high pulse; negative is a low pulse.

Example

`:DIG:LINE4:MODE TRIG, OUT` `:TRIG:DIG4:OUT:LOG NEG`	Sets line 4 mode to be a trigger output and sets the output logic of the trigger event generator to negative (asserts a low pulse).

Also see

:DIGital:LINE<n>:MODE (on page 6-34)
Digital I/O port configuration (on page 3-45)

:TRIGger:DIGital<n>:OUT:PULSewidth

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

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	10e-6 (10 µs)

Usage

`:TRIGger:DIGital<n>:OUT:PULSewidth <width>`

`:TRIGger:DIGital<n>:OUT:PULSewidth?`

`<n>`	Digital I/O trigger line (1 to 6)
`<width>`	Pulse length (0 to 100,000 s)

Details

Setting the pulse width to zero (0) seconds asserts the trigger indefinitely.

Example

`DIG:LINE1:MODE TRIG, OUT` `TRIG:DIG1:OUT:PULS 2`	Set digital line 1 to trigger out. Set the pulse to 2 s.

Also see

:DIGital:LINE<n>:MODE (on page 6-34)
:DIGital:WRITe <n> (on page 6-38)
Digital I/O port configuration (on page 3-45)

:TRIGger:DIGital<n>:OUT:STIMulus

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

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

Usage

```
:TRIGger:DIGital<n>:OUT:STIMulus <event>
:TRIGger:DIGital<n>:OUT:STIMulus?
```

<n>	Digital I/O trigger line (1 to 6)
<event>	The event to use as a stimulus; see **Details**

Details

The digital trigger pulsewidth command determines how long the trigger is asserted.

The trigger stimulus for a digital I/O line can be set to one of the trigger events that are described in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	NONE
Front-panel TRIGGER key press	DISPlay
Notify trigger block <n> (1 to 8); the trigger model generates a trigger event when it executes the notify block	NOTify<n>
A command interface trigger: • Any remote interface: *TRG • GPIB only: GET bus command • VXI-11: VXI-11 command device_trigger	COMMand
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line <n> (1 to 6)	DIGio<n>
Line edge detected on TSP-Link synchronization line <n> (1 to 3)	TSPLink<n>
Appropriate LXI trigger packet is received on LAN trigger object <n> (1 to 8)	LAN<n>
Trigger event blender <n> (1 or 2), which combines trigger events	BLENder<n>
Trigger timer <n> (1 to 4) expired	TIMer<n>
Analog trigger	ATRigger
External in trigger	EXTernal

Example

:TRIG:DIG2:OUT:STIMulus TIM3	Set the stimulus for output digital trigger line 2 to be the expiration of trigger timer 3.

Also see

Digital I/O port configuration (on page 3-45)
:DIGital:LINE<n>:STATe (on page 6-36)
:TRIGger:DIGital<n>:OUT:LOGic (on page 6-196)

:TRIGger:EXTernal:IN:CLEar

This command clears the trigger event on the external in line.

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

Usage

```
:TRIGger:EXTernal:IN:CLEar
```

Details

The event detector of a trigger enters the detected state when an event is detected. This command clears the event detector, discards the history, and clears the overrun status (sets the overrun status to false).

Example

`:TRIG:EXT:IN:CLE`	Clears the trigger event detector on I/O line 2.

Also see

[:TRIGger:EXTernal:IN:OVERrun?](#) (on page 6-200)

:TRIGger:EXTernal:IN:EDGE

This command sets the type of edge that is detected as an input on the external in line.

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

Usage

```
:TRIGger:EXTernal:IN:EDGE <detectedEdge>
:TRIGger:EXTernal:IN:EDGE?
```

`<detectedEdge>`	The trigger edge value: • Detect falling-edge triggers as inputs: `FALLing` • Detect rising-edge triggers as inputs: `RISing` • Detect either falling or rising-edge triggers as inputs: `EITHer` See **Details** for descriptions of values

Details

The input state of the external I/O line is controlled by the type of edge specified by this command.

Trigger mode values

Value	Description
FALLing	Detects falling-edge triggers as input
RISing	Detects rising-edge triggers as input
EITHer	Detects rising- or falling-edge triggers as input

Example

:TRIG:EXT:IN:EDGE RIS	Sets the external I/O to detect rising-edge triggers as input.

Also see

:TRIGger:EXTernal:IN:OVERrun?

This command returns the event detector overrun status.

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

Usage

:TRIGger:EXTernal:IN:OVERrun?

Details

This command returns the event detector overrun status as 0 (false) or 1 (true).

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

TRIG:EXT:IN:OVER?	Returns 0 if no overruns have occurred or 1 if one or more overrun have occurred for the external I/O line.

Also see

None

:TRIGger:EXTernal:OUT:LOGic

This command sets the output logic of the trigger event generator to positive or negative for the external out line.

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

Usage

```
:TRIGger:EXTernal:OUT:LOGic <logicType>
:TRIGger:EXTernal:OUT:LOGic?
```

`<logicType>`	The output logic of the trigger generator: • Assert a TTL-high pulse for output: `POSitive` • Assert a TTL-low pulse for output: `NEGative`

Details

This command sets the trigger event generator to assert a TTL pulse for output logic. Positive is a high pulse; negative is a low pulse.

Example

`*RST`
`:TRIG:EXT:IN:CLE`
`:TRIG:EXT:OUT:LOG NEG`
`:TRIG:EXT:OUT:STIM EXT`
`:TRIG:EXT:IN:EDGE FALL`
	Reset the external I/O port values to their defaults.
	Clear any event triggers on the external in line.
	Set the output logic to negative (it asserts a low pulse).
	Set the stimulus to the external input.
	Set the external input to detect a falling edge.

Also see

None

:TRIGger:EXTernal:OUT:STIMulus

This command selects the event that causes a trigger to be asserted on the external output line.

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

Usage

```
:TRIGger:EXTernal:OUT:STIMulus <event>
:TRIGger:EXTernal:OUT:STIMulus?
```

`<event>`	The event to use as a stimulus; see **Details**

Details

The trigger stimulus for the external output line can be set to one of the trigger events described in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

`:TRIG:EXT:OUT:STIMulus TIM3`	Set the stimulus for the external output to be the expiration of trigger timer 3.

Also see

:TRIGger:LAN<n>:IN:CLEar

This command clears the event detector for a LAN trigger.

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

Usage

```
:TRIGger:LAN<n>:IN:CLEar
```

<n>	The LAN event number (1 to 8) to clear

Details

The trigger event detector enters the detected state when an event is detected. This function clears a trigger event detector and discards the previous of the trigger packet.

This function clears all overruns associated with this LAN trigger.

Example

:TRIG:LAN5:IN:CLE	Clears the event detector with LAN packet 5.

Also see

:TRIGger:LAN<n>:IN:OVERrun? (on page 6-205)

:TRIGger:LAN<n>:IN:EDGE

This command sets the trigger operation and detection mode of the specified LAN event.

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

Usage

```
:TRIGger:LAN<n>:IN:EDGE <mode>
:TRIGger:LAN<n>:IN:EDGE?
```

<n>	The LAN event number (1 to 8)
<mode>	The trigger mode; see the **Details** for more information

Details

This command controls how the trigger event detector and the output trigger generator operate on the given trigger. These settings are intended to provide behavior similar to the digital I/O triggers.

LAN trigger mode values		
Mode	**Trigger packets detected as input**	**LAN trigger packet generated for output with a…**
EITHer	Rising or falling edge (positive or negative state)	negative state
FALLing	Falling edge (negative state)	negative state
RISing	Rising edge (positive state)	positive state

Example

:TRIG:LAN2:IN:EDGE FALL	Set the LAN trigger mode for event 2 to falling.

Also see

Digital I/O (on page 3-43)
TSP-Link System Expansion Interface (on page 3-102)

:TRIGger:LAN<n>:IN:OVERrun?

This command indicates the overrun status of the LAN event detector.

Type	Affected by	Where saved	Default value
Query only	TRIGger:LAN<n>:IN:CLEar	Not applicable	Not applicable

Usage

:TRIGger:LAN<n>:IN:OVERrun?

<n>	The LAN event number (1 to 8)

Details

This command indicates whether an event has been ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the synchronization line itself. It does not indicate if an overrun occurred in any other part of the trigger model, or in any other construct that is monitoring the event.

It also is not an indication of an output trigger overrun.

The trigger overrun state for the specified LAN packet is returned as 1 (true) or 0 (false).

Example

TRIG:LAN5:IN:OVER?	Checks the overrun status of a trigger on LAN5 and outputs the value, such as: 0

Also see

:TRIGger:LAN<n>:IN:CLEar (on page 6-203)

:TRIGger:LAN<n>:OUT:CONNect:STATe

This command prepares the event generator for outgoing trigger events.

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

Usage

```
:TRIGger:LAN<n>:OUT:CONNect:STATe <state>
:TRIGger:LAN<n>:OUT:CONNect:STATe?
```

`<n>`	The LAN event number (1 to 8)
`<state>`	Do not send event messages: `OFF` or `0`
	Prepare to send event messages: `ON` or `1`

Details

When this is set to `ON`, the instrument prepares the event generator to send event messages. For TCP connections, this opens the TCP connection.

The event generator automatically disconnects when either the protocol or IP address for this event is changed.

When this is set to `OFF`, for TCP connections, this closes the TCP connection.

Example

`:TRIGger:LAN1:OUT:PROTocol MULT` `:TRIGger:LAN1:OUT:CONNect:STATe ON`	Set the protocol to multicast and prepare the event generator to send event messages.

Also see

:TRIGger:LAN<n>:OUT:IP:ADDRess (on page 6-207)
:TRIGger:LAN<n>:OUT:PROTocol (on page 6-208)

:TRIGger:LAN<n>:OUT:IP:ADDRess

This command specifies the address (in dotted-decimal format) of UDP or TCP listeners.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	"0.0.0.0"

Usage

```
:TRIGger:LAN<n>:OUT:IP:ADDRess "<address>"
:TRIGger:LAN<n>:OUT:IP:ADDRess?
```

<n>	The LAN event number (1 to 8)
<address>	A string that represents the LAN address in dotted decimal notation

Details

Sets the IP address for outgoing trigger events.

After you change this setting, you must send the connect command before outgoing messages can be sent.

Example

TRIG:LAN1:OUT:IP:ADDR "192.0.32.10"	Use IP address `192.0.32.10` to connect the LAN trigger.

Also see

:TRIGger:LAN<n>:OUT:CONNect:STATe (on page 6-206)

:TRIGger:LAN<n>:OUT:LOGic

This command sets the logic on which the trigger event detector and the output trigger generator operate on the given trigger line.

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

Usage

```
:TRIGger:LAN<n>:OUT:LOGic <logicType>
:TRIGger:LAN<n>:OUT:LOGic?
```

<n>	The LAN event number (1 to 8)
<logicType>	The type of logic: • POSitive • NEGative

Example

TRIG:LAN1:OUT:LOG POS	Set the logic to positive.

Also see

None

:TRIGger:LAN<n>:OUT:PROTocol

This command sets the LAN protocol to use for sending trigger messages.

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

Usage

```
:TRIGger:LAN<n>:OUT:PROTocol
:TRIGger:LAN<n>:OUT:PROTocol?
```

<n>	The LAN event number (1 to 8)
<protocol>	The protocol to use for messages from the trigger: • TCP • UDP • MULTicast

Details

The LAN trigger listens for trigger messages on all the supported protocols. However, it uses the designated protocol for sending outgoing messages.

After you change this setting, you must re-connect the LAN trigger event generator before you can send outgoing event messages.

When multicast is selected, the trigger IP address is ignored and event messages are sent to the multicast address 224.0.23.159.

Example

:TRIG:LAN1:OUT:PROT TCP :TRIG:LAN1:OUT:CONN:STAT	Set the LAN protocol for trigger messages to be TCP and re-connect the LAN trigger event generator.

Also see

:TRIGger:LAN<n>:OUT:CONNect:STATe (on page 6-206)
:TRIGger:LAN<n>:OUT:IP:ADDRess (on page 6-207)

:TRIGger:LAN<n>:OUT:STIMulus

This command specifies events that cause this trigger to assert.

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

Usage

```
:TRIGger:LAN<n>:OUT:STIMulus <LANevent>
:TRIGger:LAN<n>:OUT:STIMulus?
```

<n>	A number specifying the trigger packet over the LAN for which to set or query the trigger source (1 to 8)
<LANevent>	The LAN event that causes this trigger to assert

Details

This attribute specifies which event causes a LAN trigger packet to be sent for this trigger. Set the event to one of the existing trigger events, which are shown in the following table.

Setting this attribute to none disables automatic trigger generation.

If any events are detected before the trigger LAN connection is sent, the event is ignored and the action overrun is set.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

`TRIG:LAN1:OUT:STIM TIM1`	Set the timer 1 trigger event as the source for the LAN packet 1 trigger stimulus.

Also see

:TRIGger:LAN<n>:OUT:CONNect:STATe (on page 6-206)

:TRIGger:LOAD "ConfigList"

This command loads a predefined trigger model configuration that uses a configuration list.

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

Usage

```
:TRIGger:LOAD  "ConfigList", "<measureConfigList>"
:TRIGger:LOAD  "ConfigList", "<measureConfigList>", <delay>
:TRIGger:LOAD  "ConfigList", "<measureConfigList>", <delay>, "<bufferName>"
:TRIGger:LOAD  "ConfigList", "<measureConfigList>", <delay>, "<bufferName>",
    <readingBlock>
```

`<measureConfigList>`	A string that contains the name of the measurement configuration list to use
`<delay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<readingBlock>`	Define a measure or digitize block for the trigger model; options are: • `ACTive`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `ACTive` is used • `MEASure`: Adds a measure block to the trigger model • `DIGitize`: Adds a digitize block to the trigger model

Details

This trigger model template incorporates a configuration list.

You must set up the configuration lists before loading the trigger model.

You can also set a delay and change the reading buffer.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

```
*RST
:SENS:CONF:LIST:CRE "MEASURE_LIST"
:SENS:CURR:RANG 1e-3
:SENSe:CONF:LIST:STOR "MEASURE_LIST"
:SENS:CURR:RANG 10e-3
:SENSe:CONF:LIST:STOR "MEASURE_LIST"
:SENS:CURR:RANG 100e-3
:SENSe:CONF:LIST:STOR "MEASURE_LIST"
:TRIG:LOAD "ConfigList", "MEASURE_LIST"
INIT
```

Set up a configuration list named `MEASURE_LIST`.
Load the configuration list trigger model, using this configuration list.
Start the trigger model.

Also see

[:SENSe[1]]:CONFiguration:LIST:CREate (on page 6-112)

:TRIGger:LOAD "DurationLoop"

This command loads a predefined trigger model configuration that makes continuous measurements for a specified amount of time.

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

Usage

```
:TRIGger:LOAD "DurationLoop", <duration>
:TRIGger:LOAD "DurationLoop", <duration>, <delay>
:TRIGger:LOAD "DurationLoop", <duration>, <delay>, "<readingBuffer>"
:TRIGger:LOAD "DurationLoop", <duration>, <delay>, "<readingBuffer>",
    <readingBlock>
```

`<duration>`	The amount of time for which to make measurements (500 ns to 100,000 s)
`<delay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
`<readingBuffer>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<readingBlock>`	Define a measure or digitize block for the trigger model; options are: • `ACTive`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `ACTive` is used • `MEASure`: Adds a measure block to the trigger model • `DIGitize`: Adds a digitize block to the trigger model

Details

When you load this predefined trigger model, you can specify amount of time to make a measurement and the length of the delay before the measurement.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

`*RST`
`SENS:FUNC "CURR"`
`TRIG:LOAD "DurationLoop", 10, 0.01`
`INIT`
	Reset the instrument.
	Set the instrument to measure DC current.
	Load the Duration Loop trigger model to make measurements for 10 s with a 10 ms delay before each measurement.
	Start the trigger model.

Also see

None

:TRIGger:LOAD "Empty"

This command resets the trigger model.

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

Usage

```
:TRIGger:LOAD "Empty"
```

Details

When you load this predefined trigger model, any blocks that have been defined in the trigger model are cleared.

Example

`TRIG:LOAD "Empty"`
`TRIG:BLOC:BUFF:CLEAR 1`
`TRIG:BLOC:MEAS 2`
`TRIG:BLOC:BRAN:COUN 3, 5, 2`
`TRIG:BLOC:DEL:CONS 4, 1`
`TRIG:BLOC:BRAN:COUN 5, 3, 2`
`INIT`
	Reset trigger model settings.
	Clear `defbuffer1` at the beginning of execution of the trigger model.
	Loop and take 5 readings.
	Delay 1 s.
	Loop three more times back to block 2.
	At the end of execution, 15 readings are stored in `defbuffer1`.

Also see

None

:TRIGger:LOAD "ExtDigTrigger"

This command loads a predefined trigger model configuration that sets up an external trigger through the digital I/O.

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

Usage

```
:TRIGger:LOAD "ExtDigTrigger", <digInLine>, <digOutLine>, <count>
:TRIGger:LOAD "ExtDigTrigger", <digInLine>, <digOutLine>, <count>, <delay>
:TRIGger:LOAD "ExtDigTrigger", <digInLine>, <digOutLine>, <count>, <delay>,
    "<bufferName>"
:TRIGger:LOAD "ExtDigTrigger", <digInLine>, <digOutLine>, <count>, <delay>,
    "<bufferName>", <readingBlock>
```

`<digInLine>`	The digital input line (1 to 6) or external input line (7); also the event that the trigger model will wait on in block 1
`<digOutLine>`	The digital output line (1 to 6) or external output line (7)
`<count>`	The number of measurements the instrument will make
`<delay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
`<bufferName>`	The reading buffer that will store the measurement data, default is `defbuffer1`
`<readingBlock>`	Define a measure or digitize block for the trigger model; options are: • `ACTive`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `ACTive` is used • `MEASure`: Adds a measure block to the trigger model • `DIGitize`: Adds a digitize block to the trigger model

Details

This trigger model waits for a digital input or external trigger input event to occur, makes a measurement, and issues a notify event. If a digital output line is selected, a notify event asserts a digital output line. A notify event asserts the external output line regardless of the line settings. You can set the line to `7` to assert only the external output line, or to another setting to assert both a digital output line and the external output line.

Example

```
:TRIGger:LOAD "ExtDigTrigger", 7, 2, 10, 0.001, "defbuffer1", ACT
```

Set up the template to use the external in line and wait for a pulse from the external in to trigger measurements.
Pulse digital out line 2 when the measurement is complete. The external output line is also pulsed.
Make 10 measurements, with a delay of 1 ms before each measurement.
Store the measurements in `defbuffer1`.

Also see

None

:TRIGger:LOAD "GradeBinning"

This command loads a predefined trigger model configuration that sets up a grading operation.

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

Usage

```
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
    <limit3Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
    <limit3Pattern>, <limit4High>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
    <limit3Pattern>, <limit4High>, <limit4Low>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>,  <limit3High>, <limit3Low>,
    <limit3Pattern>, <limit4High>, <limit4Low>, <limit4Pattern>
:TRIGger:LOAD "GradeBinning", <components>, <startInLine>, <startDelay>,
    <endDelay>, <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>,
    <limit2High>, <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>,
    <limit3Pattern>, <limit4High>, <limit4Low>, <limit4Pattern>, "<bufferName>"
```

<components>	The number of components to measure (1 to 268,435,455)
<startInLine>	The input line that starts the test; 5 for digital line 5, 6 for digital line 6, or 7 for external in; default is 5
<startDelay>	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
<endDelay>	The delay time after the measurement (167 ns to 10,000 s); default is 0 for no delay

`<limitxHigh>`	*x* is limit 1, 2, 3, or 4; the upper limit that the measurement is compared against
`<limitxLow>`	*x* is 1, 2, 3, or 4; the lower limit that the measurement is compared against
`<limit1Pattern>`	The bit pattern that is sent when the measurement fails limit 1; range 1 to 15; default is 1
`<limit2Pattern>`	The bit pattern that is sent when the measurement fails limit 2; range 1 to 15; default is 2
`<limit3Pattern>`	The bit pattern that is sent when the measurement fails limit 3; range 1 to 15; default is 4
`<limit4Pattern>`	The bit pattern that is sent when the measurement fails limit 4; range 1 to 15; default is 8
`<allPattern>`	The bit pattern that is sent when all limits have passed; 1 to 15; default is 15
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This trigger model template allows you to grade components and place them into up to four bins, based on the comparison to limits.

To set a limit as unused, set the high value for the limit to be less than the low limit.

All limit patterns and the pass pattern are sent on digital I/O lines 1 to 4, where 1 is the least significant bit.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

For a detailed example, see the section in the *Model DMM7510 User's Manual* named "Grading and binning resistors."

Also see

None

:TRIGger:LOAD "Keithley2001"

This command loads a predefined trigger model configuration that emulates a Keithley Instruments 2001 trigger model.

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

Usage

```
:TRIGger:LOAD  "Keithley2001", <arm1Bypass>, <arm1Source>, <arm1Count>,
    <arm2Bypass>, <arm2Source>, <arm2Count>, <arm2Delay>, <trigBypass>,
    <trigSource>, <trigCount>, <trigDelay>
```

`<arm1Bypass>`	Bypass Arm 1: `ON` (`1`) Do not bypass Arm 1: `OFF` (`0`)
`<arm1Source>`	The event that triggers Arm 1; see **Details**
`<arm1Count>`	The number of times of times to repeat Arm 1
`<arm2Bypass>`	Bypass Arm 2: `ON` Do not bypass Arm 2: `OFF`
`<arm2Source>`	The event that triggers Arm 2; see **Details**
`<arm2Count>`	The number of times of times to repeat Arm 2
`<arm2Delay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
`<trigBypass>`	Bypass the trigger layer: `ON` Do not bypass the trigger layer: `OFF`
`<trigSource>`	The event that triggers the trigger layer; see **Details**
`<trigCount>`	The number of times of times to repeat the trigger layer
`<trigDelay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay

Details

If the trigger layer is not bypassed, the External In/Out terminal is asserted. The arm layers do not assert the external in/out terminal.

You can use this template to emulate other trigger models if you use only one of the arm layers. Set the other arm layers to a source of `NONE`, a count of `1` and a delay of `0`.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Trigger events	
Event description	**Event constant**
No trigger	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8) generates a trigger event when the trigger model executes it	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
External in trigger	`EXTernal`

Example

Refer to the application notes for the Model DMM7510 on the Keithley Instruments website *http://www.keithley.com* for an example with additional detail about this command.

Also see

None

:TRIGger:LOAD "LoopUntilEvent"

This command loads a predefined trigger model configuration that makes continuous measurements until the specified event occurs.

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

Usage

```
:TRIGger:LOAD "LoopUntilEvent", <eventNumber>, <position>
:TRIGger:LOAD "LoopUntilEvent", <eventNumber>, <position>, <delay>
:TRIGger:LOAD "LoopUntilEvent", <eventNumber>, <position>, <delay>,
    "<readingBuffer>"
:TRIGger:LOAD "LoopUntilEvent", <eventNumber>, <position>, <delay>,
    "<readingBuffer>", <readingBlock>
```

`<eventNumber>`	The event that ends infinite triggering or readings set to occur before the trigger; see **Details**
`<position>`	The number of readings to make in relation to the size of the reading buffer; enter as a percentage
`<delay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
`<readingBuffer>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<readingBlock>`	Define a measure or digitize block for the trigger model; options are: • `ACTive`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `ACTive` is used • `MEASure`: Adds a measure block to the trigger model • `DIGitize`: Adds a digitize block to the trigger model

Details

The source event is the event that ends infinite triggering or ends readings set to occur before the trigger and start post-trigger readings. The trigger model makes readings until it detects the source event. After the source event, it makes a finite number of readings, based on the setting of the trigger position.

The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled. For example, if this is set to 75 for a reading buffer that holds 10,000 readings, the trigger model makes 2500 readings after it detects the source event. There will be 7500 pre-trigger readings and 2500 post-trigger readings.

The instrument makes two sets of readings. The first set is made until the trigger event occurs. The second set is made after the trigger event occurs, up to the number of readings calculated by the position parameter.

You cannot have the source event set at none when you run this predefined trigger model.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Trigger events	
Event description	**Event constant**
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

| ```
*RST
SENS:FUNC "CURR"
TRIG:LOAD "LoopUntilEvent", DISP, 25
INIT
``` | Reset the instrument.<br>Set the instrument to measure DC current.<br>Set the LoopUntilEvent trigger model to make measurements until the front-panel TRIGGER key is pressed, then make measurements that constitute 75 % of the reading buffer.<br>Start the trigger model. |

Also see

None

:TRIGger:LOAD "SimpleLoop"

This command loads a predefined trigger model configuration.

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

Usage

```
:TRIGger:LOAD "SimpleLoop", <count>
:TRIGger:LOAD "SimpleLoop", <count>, <delay>
:TRIGger:LOAD "SimpleLoop", <count>, <delay>, "<bufferName>"
:TRIGger:LOAD "SimpleLoop", <count>, <delay>, "<bufferName>", <readingBlock>
```

`<count>`	The number of measurements the instrument will make
`<delay>`	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
`<readingBlock>`	Define a measure or digitize block for the trigger model; options are: • `ACTive`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `ACTive` is used • `MEASure`: Adds a measure block to the trigger model • `DIGitize`: Adds a digitize block to the trigger model

Details

This command sets up a loop that sets a delay, makes a measurement, and then repeats the loop the number of times you define in the count parameter.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

```*RST```
 ```SENS:FUNC "CURR"```
 ```SENS:CURR:RANG:AUTO ON```
 ```TRIG:LOAD "SimpleLoop", 10```
 ```INIT```
 ```*WAI```
 ```TRAC:DATA? 1, 10, "defbuffer1", READ, REL```
	Reset the instrument and set it to measure current with automatic range setting.
	Set a simple trigger loop with a count of 10.
	Start the trigger model.
	Postpone execution of subsequent commands until all previous commands are finished.
	Read data and return the reading and relative time.

Also see

None

:TRIGger:LOAD "SortBinning"

This command loads a predefined trigger model configuration that sets up a sorting operation.

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

Usage

```
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
    <limit4High>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
    <limit4High>, <limit4Low>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
    <limit4High>, <limit4Low>, <limit4Pattern>
:TRIGger:LOAD "SortBinning", <components>, <startInLine>, <startDelay>, <endDelay>,
    <limit1High>, <limit1Low>, <limit1Pattern>, <allPattern>, <limit2High>,
    <limit2Low>, <limit2Pattern>, <limit3High>, <limit3Low>, <limit3Pattern>,
    <limit4High>, <limit4Low>, <limit4Pattern>, "<bufferName>"
```

<components>	The number of components to measure
<startInLine>	The input line that starts the test; 5 for digital line 5, 6 for digital line 6, or 7 for external in; default is 5
<startDelay>	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
<endDelay>	The delay time after the measurement (167 ns to 10,000 s); default is 0 for no delay

`<limitxHigh>`	x is limit 1, 2, 3, or 4; the upper limit that the measurement is compared against
`<limitxLow>`	x is 1, 2, 3, or 4; the lower limit that the measurement is compared against
`<limit1Pattern>`	The bit pattern that is sent when the measurement passes limit 1; range 1 to 15; default is 1
`<limit2Pattern>`	The bit pattern that is sent when the measurement passes limit 2; range 1 to 15; default is 2
`<limit3Pattern>`	The bit pattern that is sent when the measurement passes limit 3; range 1 to 15; default is 4
`<limit4Pattern>`	The bit pattern that is sent when the measurement passes limit 4; range 1 to 15; default is 8
`<allPattern>`	The bit pattern that is sent when all limits have failed; 1 to 15; default is 15
`<bufferName>`	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used

Details

This trigger model template allows you to sort components and place them into up to four bins, based on the comparison to limits.

To set a limit as unused, set the high value for the limit to be less than the low limit.

All limit patterns and the all fail pattern are sent on digital I/O lines 1 to 4, where 1 is the least significant bit.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

For a detailed example, see the section in the *Model DMM7510 User's Manual* named "Grading and binning resistors."

Also see

None

:TRIGger:STATe?

This command returns the present state of the trigger model.

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

Usage

```
:TRIGger:STATe?
```

Details

This command returns the state of the trigger model. The instrument checks the state of a started trigger model every 100 ms.

This command returns the trigger state and the block that the trigger model last executed.

The trigger model states are:

- Idle: The trigger model is stopped.
- Running: The trigger model is running.
- Waiting: The trigger model has been in the same wait block for more than 100 ms.
- Empty: The trigger model is selected, but no blocks are defined.
- Building: Blocks have been added.
- Failed: The trigger model is stopped because of an error.
- Aborting: The trigger model is stopping because of a user request.
- Aborted: The trigger model is stopped because of a user request.

Example

`:TRIG:STAT?`	An example output if the trigger model is inactive and ended at block 9 would be: `IDLE;IDLE;9`

Also see

None

:TRIGger:TIMer<n>:CLEar

This command clears the timer event detector and overrun indicator for the specified trigger timer number.

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

Usage

```
:TRIGger:TIMer<n>:CLEar
```

<n>	Trigger timer number (1 to 4)

Details

This command sets the timer event detector to the undetected state and resets the overrun indicator.

Example

`:TRIG:TIM1:CLEar`	Clears trigger timer 1.

Also see

:TRIGger:TIMer<n>:COUNt (on page 6-224)
:TRIGger:TIMer<n>:STARt:OVERrun? (on page 6-228)

:TRIGger:TIMer<n>:COUNt

This command sets the number of events to generate each time the timer generates a trigger event or is enabled as a timer or alarm.

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

Usage

```
:TRIGger:TIMer<n>:COUNt <count>
:TRIGger:TIMer<n>:COUNt?
```

<n>	Trigger timer number (1 to 4)
<count>	The number of times to repeat the trigger (0 to 1,048,575)

Details

If *count* is set to a number greater than 1, the timer automatically starts the next trigger timer delay at the expiration of the previous delay.

Set *count* to zero (0) to cause the timer to generate trigger events indefinitely.

If you use the trigger timer with a trigger model, make sure the count value is the same or more than any count values expected in the trigger model.

Example 1

`TRIG:TIM2:COUN 4`	Set the number of events to generate for trigger timer 2 to four.

Example 2

```
*RST
TRIG:TIM4:DEL 0.5
TRIG:TIM4:STAR:STIM NOT8
TRIG:TIM4:STAR:GEN OFF
TRIG:TIM4:COUN 20
TRIG:TIM4:STAT ON

TRIG:LOAD "EMPTY"
TRIG:BLOC:BUFF:CLEAR 1, "defbuffer1"
TRIG:BLOC:NOT 2, 8
TRIG:BLOC:WAIT 3, TIM4
TRIG:BLOC:MEAS 4, "defbuffer1"
TRIG:BLOC:BRAN:COUN 5, 20, 3
INIT
TRAC:ACT? "defbuffer1"
```

Set trigger timer 4 to have a 0.5 s delay.
Set the stimulus for trigger timer 4 to be the notify 8 event.
Set the trigger timer 4 stimulus to off.
Set the timer event to occur when the timer delay elapses.
Set the trigger timer 4 count to 20.
Enable trigger timer 4.

Clear the trigger model.
Set trigger model block 1 to clear the buffer.
Set trigger model block 2 to generate the notify 8 event.
Set trigger model block 3 to wait for trigger timer 4 to occur.
Set trigger model block 4 to make a measurement and store it in default buffer 1.
Set trigger model block 5 to repeat the trigger model 20 times, starting at block 3.
Start the trigger model.
Output the number of entries in default buffer 1.

Output:
```
20
```

Also see

:TRIGger:TIMer<n>:DELay

This command sets and reads the timer delay.

Type	Affected by	Where saved	Default value
Command and query	Recall settings Instrument reset Power cycle	Save settings	10e-6 (10 µs)

Usage

```
:TRIGger:TIMer<n>:DELay <interval>
:TRIGger:TIMer<n>:DELay?
```

<n>	Trigger timer number (1 to 4)
<interval>	Delay interval (8e-6 s to 100,000 s)

Details

Each time the timer is triggered, it uses this delay period.

Reading this command returns the delay interval that will be used the next time the timer is triggered.

Example

TRIG:TIM2:DEL 50E-6	Set trigger timer 2 to delay for 50 µs.

Also see

None

:TRIGger:TIMer\<n>:STARt:FRACtional

This command configures an alarm or a time in the future when the timer will start.

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

Usage

```
:TRIGger:TIMer<n>:STARt:FRACtional <time>
:TRIGger:TIMer<n>:STARt:FRACtional?
```

`<n>`	Trigger timer number (1 to 4)
`<time>`	The time in fractional seconds (0 to 2,147,483,647 ns)

Details

This command configures the alarm of the timer.

When the timer is enabled, the timer starts immediately if the timer is configured for a start time in the past or if it is in the future.

Example

``` TRIG:TIM1:STAR:SEC 60 TRIG:TIM1:START:FRAC 0.5 TRIG:TIM1:STAT ON ```	Set the timer for 60.5 s. Enable the trigger timer for timer 1.

Also see

:TRIGger:TIMer\<n>:STARt:SEConds (on page 6-229)
:TRIGger:TIMer\<n>:STATe (on page 6-231)

:TRIGger:TIMer<n>:STARt:GENerate

This command specifies when timer events are generated.

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

Usage

```
:TRIGger:TIMer<n>:STARt:GENerate <state>
:TRIGger:TIMer<n>:STARt:GENerate?
```

<n>	Trigger timer number (1 to 4)
<state>	Generate a timer event when the timer delay elapses: OFF or 0 Generate a timer event when the timer starts and when the delay elapses: ON or 1

Details

When this is set to on, a trigger event is generated immediately when the timer is triggered.

When it is set to off, a trigger event is generated when the timer elapses. This generates the event TIMER*N*.

Example

TRIG:TIM3:STAR:GEN ON	Set trigger timer 3 to generate an event when the timer starts and when the timer delay elapses.

Also see

None

:TRIGger:TIMer<n>:STARt:OVERrun?

This command indicates if an event was ignored because of the event detector state.

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

Usage

```
:TRIGger:TIMer<n>:STARt:OVERrun?
```

<n>	Trigger timer number (1 to 4)

Details

This command indicates if an event was ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the timer itself. It does not indicate if an overrun occurred in any other part of the trigger model or in any other construct that is monitoring the delay completion event. It also is not an indication of a delay overrun.

This returns 0 if there is no overrun or 1 if there is an overrun.

Example

TRIG:TIM1:STAR:OVER?	Checks the overrun status on trigger timer 1.

Also see

None

:TRIGger:TIMer<n>:STARt:SEConds

This command configures an alarm or a time in the future when the timer will start.

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

Usage

:TRIGger:TIMer<n>:STARt:SEConds <time>
:TRIGger:TIMer<n>:STARt:SEConds?

<n>	Trigger timer number (1 to 4)
<time>	The time: 0 to 2,147,483,647 s

Details

This command configures the alarm of the timer.

When the timer is enabled, the timer starts immediately if the timer is configured for a start time that has passed.

Example

TRIG:TIM1:STAR:SEC 60 TRIG:TIM1:START:FRAC 0.5 TRIG:TIM1:STAT ON	Set the timer for 60.5 s. Enable the trigger timer for timer 1.

Also see

:TRIGger:TIMer\<n>:STARt:STIMulus

This command describes the event that starts the trigger timer.

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

Usage

```
:TRIGger:TIMer<n>:STARt:STIMulus <event>
:TRIGger:TIMer<n>:STARt:STIMulus?
```

`<n>`	Trigger timer number (1 to 4)
`<event>`	The event that starts the trigger timer

Details

Set this attribute any trigger event to start the timer when that event occurs.

Set this attribute to zero (0) to disable event processing and use the timer as a timer or alarm based on the start time.

Trigger events are described in the table below.

Trigger events	
Event description	**Event constant**
No trigger event	`NONE`
Front-panel TRIGGER key press	`DISPlay`
Notify trigger block `<n>` (1 to 8); the trigger model generates a trigger event when it executes the notify block	`NOTify<n>`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`COMMand`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line `<n>` (1 to 6)	`DIGio<n>`
Line edge detected on TSP-Link synchronization line `<n>` (1 to 3)	`TSPLink<n>`
Appropriate LXI trigger packet is received on LAN trigger object `<n>` (1 to 8)	`LAN<n>`
Trigger event blender `<n>` (1 or 2), which combines trigger events	`BLENder<n>`
Trigger timer `<n>` (1 to 4) expired	`TIMer<n>`
Analog trigger	`ATRigger`
External in trigger	`EXTernal`

Example

`*RST`
`DIG:LINE1:MODE TRIG,IN`
`DIG:LINE2:MODE TRIG,OUT`
`TRIG:TIM1:DEL 35e-3`
`TRIG:TIM1:STAR:STIM DIG1`
`TRIG:DIG2:OUT:STIM TIM1`
	Reset the instrument to default settings. Set digital I/O line 1 for use as a trigger input. Set digital I/O line 2 for use as a trigger output.
	Set timer 1 to delay 35 ms.
	Set timer 1 to start delaying once the digital I/O 1 event is detected.
	Set digital I/O line 2 to output a pulse once the timer 1 event is detected.

Also see

None

:TRIGger:TIMer<n>:STATe

This command enables the trigger timer.

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

Usage

`:TRIGger:TIMer<n>:STATe <state>`
`:TRIGger:TIMer<n>:STATe?`

`<n>`	Trigger timer number (1 to 4)
`<state>`	Disable the trigger timer: `OFF` or `0` Enable the trigger timer: `ON` or `1`

Details

When this command is set to on, the timer performs the delay operation.

When this command is set to off, there is no timer on the delay operation.

You must enable a timer before it can use the delay settings or the alarm configuration. For expected results from the timer, it is best to disable the timer before changing a timer setting, such as delay or start seconds.

To use the timer as a simple delay or pulse generator with digital I/O lines, make sure the timer start time in seconds and fractional seconds is configured for a time in the past. To use the timer as an alarm, configure the timer start time in seconds and fractional seconds for the desired alarm time.

Example 1

`DIG:LINE3:MODE TRIG,OUT`
`TRIG:DIG3:OUT:STIM TIM2`
`SYSTem:TIME?`
`TRIG:TIM2:START:SECONDS <current time> + 60`
`TRIG:TIM2:STAT ON`
	To configure timer 2 for an alarm to fire 1 minute from now and output a pulse on digital I/O line 3, query to get the current time. Add 60 s to that value and use that to configure the start seconds. Enable the timer.

Example 2

`*RST`
`DIG:LINE5:MODE TRIG,OUT`
`TRIG:DIG5:OUT:STIM TIM3`
`TRIG:TIM3:DEL 3e-3`
`TRIG:TIM3:COUNT 5`
`TRIG:TIM3:STAT ON`
	Configure timer 3 to generate 5 pulses on digital I/O line 5 that are 3 ms apart.

Example 3

| ```
*RST
DIG:LINE3:MODE TRIG,IN
DIG:LINE5:MODE TRIG,OUT
TRIG:DIG5:OUT:STIM TIM3
TRIG:TIM3:DEL 3e-3
TRIG:TIM3:COUNT 5
TRIG:TIM3:START:STIM DIG3
TRIG:TIM3:STAT ON
``` | Configure timer 3 to generate 5 pulses on digital I/O line 5 that are 3 ms apart when a digital input is detected on digital line 3. |

Also see

None

Introduction to TSP commands

Introduction to TSP operation

Instruments that are Test Script Processor (TSP®) enabled operate like conventional instruments by responding to a sequence of commands sent by the controller. You can send individual commands to the TSP-enabled instrument the same way you would when using any other instrument.

Unlike conventional instruments, TSP-enabled instruments can execute automated test sequences independently, without an external controller. You can load a series of TSP commands into the instrument using a remote computer or the front-panel port with a USB flash drive. You can store these commands as a script that can be run later by sending a single command message to the instrument.

You do not have to choose between using conventional control or script control. You can combine these forms of instrument control in the way that works best for your test application.

Controlling the instrument by sending individual command messages

The simplest method of controlling an instrument through the communication interface is to send it a message that contains remote commands. You can use a test program that resides on a computer (the controller) to sequence the actions of the instrument.

TSP commands can be function-based or attribute-based. Function-based commands are commands that control actions or activities. Attribute-based commands define characteristics of an instrument feature or operation.

Constants and enumerated types are commands that represent fixed values.

Functions

Function-based commands control actions or activities. A function-based command performs an immediate action on the instrument.

Each function consists of a function name followed by a set of parentheses ( ). You should only include information in the parentheses if the function takes a parameter. If the function takes one or more parameters, they are placed between the parentheses and separated by commas.

Example 1

| ```
beeper.beep(0.5, 2400)
delay(0.250)
beeper.beep(0.5, 2400)
``` | Emit a double-beep at 2400 Hz. The sequence is 0.5 s on, 0.25 s off, 0.5 s on. |

Example 2

You can use the results of a function-based command directly or assign variables to the results for later access. The following code defines x and prints it.

| ```
x = math.abs(-100)
print(x)
``` | Output:<br>`100` |

Attributes

Attribute-based commands are commands that set the characteristics of an instrument feature or operation. For example, some characteristics of TSP-enabled instruments are the model number (`localnode.model`) and the brightness of the front-panel display (`display.lightstate`).

Attributes can be read-only, read-write, or write-only. They can be used as a parameter of a function or assigned to another variable.

To set the characteristics, attribute-based commands define a value. For many attributes, the value is in the form of a number, enumerated type, or a predefined constant.

Example 1: Set an attribute using a number

| ```
testData = buffer.make(500)
testData.capacity = 600
``` | Use a function to create a buffer named `testData` with a capacity of 500, then use the *bufferVar*.`capacity` attribute to change the capacity to 600. |

Example 2: Set an attribute using an enumerated type

| ```
display.lightstate = display.STATE_LCD_75
``` | This attribute controls the brightness of the front-panel display and buttons. Setting this attribute to `display.STATE_LCD_75` sets the brightness of the display and buttons to 75 % of full brightness. |

Example 3: Set an attribute using a constant

| ```
format.data = format.REAL64
``` | Using the constant `REAL64` sets the print format to double precision floating point format. |

Reading an attribute

To read an attribute, you can use the attribute as the parameter of a function or assign it to another variable.

Example 1: Read an attribute using a function

`print(display.lightstate)`	Reads the status of the light state by passing the attribute to the print function. If the display light state is set to 50 %, the output is: `display.STATE_LCD_50`

Example 2: Read an attribute using a variable

`light = display.lightstate` `print(light)`	This reads the light state by assigning the attribute to a variable named `light`. If the display light state is set to 25 %, the output is: `display.STATE_LCD_25`

Queries

Test Script Processor (TSP®) enabled instruments do not have inherent query commands. Like any other scripting environment, the `print()` and `printnumber()` commands generate output in the form of response messages. Each `print()` command creates one response message.

Example

`x = 10` `print(x)`	Example of an output response message: `1.00000e+01` Note that your output may be different if you set your ASCII precision setting to a different value.

USB flash drive path

You can use the file commands to open and close directories and files, write data, or to read a file on an installed USB flash drive.

The root folder of the USB flash drive has the absolute path:

`"/usb1/"`

Information on scripting and programming

If you need information about using scripts with your TSP-enabled instrument, see <u>Fundamentals of scripting for TSP</u> (on page 7-4).

If you need information about using the Lua programming language with the instrument, see <u>Fundamentals of programming for TSP</u> (on page 7-11).

Fundamentals of scripting for TSP

NOTE
Though it can improve your process to use scripts, you do not have to create scripts to use the instrument. Most of the examples in the documentation can be run by sending individual command messages. The next few sections of the documentation describe scripting and programming features of the instrument. You only need to review this information if you are using scripting and programming.

Scripting helps you combine commands into a block of code that the instrument can run. Scripts help you communicate with the instrument more efficiently.

Scripts offer several advantages compared to sending individual commands from the host controller (computer):

- Scripts are easier to save, refine, and implement than individual commands.
- The instrument performs more quickly and efficiently when it processes scripts than it does when it processes individual commands.
- You can incorporate features such as looping and branching into scripts.
- Scripts allow the controller to perform other tasks while the instrument is running a script, enabling some parallel operation.
- Scripts eliminate repeated data transfer times from the controller.

In the instrument, the Test Script Processor (TSP®) scripting engine processes and runs scripts.

This section describes how to create, load, modify, and run scripts.

What is a script?

A script is a collection of instrument control commands and programming statements. Scripts that you create are referred to as **user scripts**.

Your scripts can be interactive. Interactive scripts display messages on the front panel of the instrument that prompt the operator to enter parameters.

Run-time and nonvolatile memory storage of scripts

Scripts are loaded into the run-time environment of the instrument. From there, they can be stored in nonvolatile memory in the instrument.

The run-time environment is a collection of global variables, which include scripts, that the user has defined. A global variable can be used to store a value while the instrument is turned on. When you create a script, the instrument creates a global variable with the same name so that you can reference the script more conveniently. After scripts are loaded into the run-time environment, you can run and manage them from the front panel of the instrument or from a computer. Information in the run-time environment is lost when the instrument is turned off.

Nonvolatile memory is where information is stored even when the instrument is turned off. Save scripts to nonvolatile memory to save them even if the power is cycled. The scripts that are in nonvolatile memory are loaded into the run-time environment when the instrument is turned on.

Scripts are placed in the run-time environment at the following times:

- When they are run.
- When they are loaded over a remote command interface.
- When the instrument is turned on (if they are stored in nonvolatile memory).

For detail on the amount of available memory, see Memory considerations for the run-time environment (on page 7-40).

> **NOTE**
>
> If you make changes to a script in the run-time environment, the changes are lost when the instrument is turned off. To save the changes, you must save them to nonvolatile memory. See Saving a script to nonvolatile memory (on page 7-8).

What can be included in scripts?

Scripts can include combinations of TSP commands and Lua code. TSP commands instruct the instrument to do one thing and are described in the command reference (see TSP commands (on page 8-7)). Lua is a scripting language that is described in Fundamentals of programming for TSP (on page 7-11).

Working with scripts

This section describes the basics of working with scripts.

You can create and manage scripts from the front panel or over a remote interface. Scripts can be saved in the instrument, on a computer, or on a USB flash drive.

Tools for managing scripts

You can use any of the following tools to manage scripts:

- The front-panel menu options and USB flash drive. For information, refer to Saving setups (on page 2-148).
- The front-panel interface options in the Scripts menu. For information, refer to the following sections.
- Messages sent to the instrument. For information, see Load a script by sending commands over a remote interface (on page 7-7).
- Keithley Instruments Test Script Builder (TSB) software (available for download on the Keithley Instruments support website (*http://www.keithley.com/support*)). For more information, see Creating a new TSP project (on page 7-34).
- Your own development tool or program.

Script rules

You can have as many scripts as needed in the instrument. The only limitation is the amount of memory available to the run-time environment.

When a script is loaded into the run-time environment, a global variable with the same name as the script is created to reference the script.

Important points regarding scripts:

- Each script must have a unique name.
- Script names must not contain spaces.
- If you load a new script with the same name as an existing script, an error event message is generated. You must delete the existing script before you create a new script with the same name.
- If you revise a script and save it to the instrument with a new name, the previously loaded script remains in the instrument with the original name.
- You can save scripts to nonvolatile memory in the instrument. Saving a script to nonvolatile memory allows the instrument to be turned off without losing the script. See Saving a script to nonvolatile memory (on page 7-8).

Loading a script into the instrument

You can load scripts from the front-panel display by copying them from a USB flash drive. You can also load them over a remote interface using `loadscript` commands.

Loading a script using a USB flash drive

After loading a script onto a USB flash drive, you can copy the script using options on the front-panel display.

To load a script using a USB flash drive:

1. Insert the USB flash drive into the USB port on the front panel.
2. Press the **MENU** key.
3. Under Scripts, select **Manage**. The MANAGE SCRIPTS window is displayed.
4. In the USB Scripts list, select the script you want to copy from the USB flash drive.
5. Select **<**. The file is transferred to the USB flash drive, and the corresponding filename is displayed in the Internal Scripts box.

Load a script by sending commands over a remote interface

To load a script over the remote interface, you can use the `loadscript` and `endscript` commands.

Normally, when the instrument receives a command, it runs the command immediately. When the instrument receives the `loadscript` command, the instrument starts collecting subsequent messages instead of running them immediately.

The `endscript` command tells the instrument to stop collecting messages. It then compiles the collection of messages into a script. The script is stored as a function. This script is loaded into the run-time environment — you need to save it to store it in the instrument.

To load a script:

Send the `loadscript` command with a script name. This tells the instrument to start collecting messages for the function named `testInfo`:

```
loadscript testInfo
```

Send the commands for the script; this example displays text on the USER swipe screen when the script is run:

```
display.settext(display.TEXT1, "Batch 233")
display.settext(display.TEXT2, "Test Information")
display.changescreen(display.SCREEN_USER_SWIPE)
```

Send the command that tells the instrument that the script is complete:

```
endscript
```

Run the script by sending the script name followed by `()`:

```
testInfo()
```

The USER swipe screen on the front panel is displayed and shows the text "Batch 233 Test Information" when you run this script.

To save the script to nonvolatile memory, send the command:

```
testInfo.save()
```

To load a script by sending commands:

1. Send the command `loadscript` *scriptName*, where *scriptName* is the name of the script. The name must be a legal Lua variable name.
2. Send the commands that need to be included in the script.
3. Send the command `endscript`.
4. You can now run the script. Send the script name followed by `()`. For more information, see Running scripts using a remote interface (on page 7-8).

Running scripts using the front-panel interface

To run a script from the front-panel interface:

1. Press the **MENU** key.
2. Under Scripts, select **Run**. The RUN SCRIPTS window is displayed.
3. From the Available Scripts list, select the script you want to run.
4. Select **Run Selected**.

Running scripts using a remote interface

You can run any script using *scriptVar*.run(). Replace *scriptVar* with the name of a script that is in nonvolatile or run-time memory.

Saving a script to nonvolatile memory

You can save scripts to nonvolatile memory. To keep a script through a power cycle, you must save the script to nonvolatile memory.

To save a script to nonvolatile memory:

1. Create and load a script.
2. Send the command *scriptVar*.save(), where *scriptVar* is the name of the script.

Example: Save a user script to nonvolatile memory

test1.save()	Assume a script named test1 has been loaded. test1 is saved into nonvolatile memory.

Saving a script to a USB flash drive

You can save scripts to a USB flash drive.

To save a script to an external USB flash drive:

1. Load a script.
2. Insert a USB flash drive into the USB port on the front panel.
3. Send the command *scriptVar*.save("/usb1/*filename*.tsp"), where *scriptVar* is the variable referencing the script and *filename* is the name of the file.

Rename a script

To rename a script in the runtime environment:

1. Load the script into the runtime environment with a different name.
2. Delete the previous version of the script.

To rename a script in nonvolatile memory:

Send the commands:

```
scriptVar = script.load(file)
scriptVar.save()
```

Where:

scriptVar is the name of variable that references the script

file is the path and file name of the script file to load

For example, to load a script named test8 from the USB flash drive and save it to nonvolatile memory, send the commands:

```
test8 = script.load("/usb1/test8.tsp")
test8.save()
```

NOTE

If the new name is the same as a name that is already used for a script, an event message is displayed and the script is not saved.

Retrieve a user script from the instrument

You can review user scripts that are in the nonvolatile memory of the instrument and retrieve them.

To see a list of scripts from the front-panel interface:
1.   Press the **MENU** key.
2.   Under Scripts, select **Manage**. The MANAGE SCRIPTS window is displayed.

The scripts are listed in the Internal Scripts list. To see the contents of the script, you can copy them to a USB flash drive. You can read the scripts with a text editor. See Saving a script to a USB flash drive (on page 7-8).

If you are using the TSP command set, you can also retrieve a list of scripts by sending the following code:

```
for name in script.user.catalog() do
   print(name)
end
```

To retrieve the content of a script, use *scriptVar*.source, where *scriptVar* is the name of the script you want to retrieve. For example, to retrieve a script named contactTest, you would send:

```
print(contactTest.source)
```

The command is returned as a single string. The loadscript and endscript keywords are not included.

Deleting a user script using a remote interface

Deleting a user script deletes the script from the instrument.

To delete a script from the instrument:

Send the command:

```
script.delete("name")
```

Where: *name* is the user-defined name of the script.

Example: Delete a user script

script.delete("test8")	Delete a user script named test8 from the instrument.

Power up script

The power up script runs automatically when the instrument is powered on. To create a power up script, save a new script and name it autoexec. The autoexec script is automatically saved to nonvolatile memory. See Saving a script to nonvolatile memory (on page 7-8).

To set up the power up script from the front panel:

1. Press the **MENU** key.
2. Under Scripts, select **Run**.
3. Select **Copy to Power Up**. A dialog box confirms that the script was copied.
4. Select **OK**.

To save the power up script using remote commands:

Send the command:

```
autoexec.save()
```

If an autoexec script already exists, you need to delete it. Send the command:

```
script.delete("autoexec")
```

Commands that cannot be used in scripts

You cannot use the following commands in scripts:

- abort
- endflash
- endscript
- flash
- loadscript
- loadandrunscript
- login
- logout
- prevflash

Common commands that cannot be used in scripts are shown in the following table with equivalent commands that can be used.

Unavailable commands with TSP equivalents

Common commands	TSP equivalent commands
`*CLS`	`eventlog.clear() status.clear()`
`*ESE`	`status.standard.enable`
`*ESE?`	`print(status.standard.enable)`
`*ESR?`	`print(status.standard.event)`

`*IDN?`	`print(localnode.model)`
	`print(localnode.serialno)`
	`print(localnode.version)`

`*LANG`	No equivalent
`*LANG?`	No equivalent
`*OPC`	`opc()`
`*OPC?`	`waitcomplete() print([[1]])`
`*RST`	`reset()`
`*SRE`	`status.request_enable`
`*SRE?`	`print(status.request_enable)`
`*STB?`	`print(status.condition)`
`*TRG`	No equivalent
`*TST?`	`print([[0]])`
`*WAI`	`waitcomplete()`

Fundamentals of programming for TSP

To conduct a test, a computer (controller) is programmed to send sequences of commands to an instrument. The controller orchestrates the actions of the instrumentation. The controller is typically programmed to request measurement results from the instrumentation and make test sequence decisions based on those measurements.

To take advantage of the advanced features of the instrument, you can add programming commands to your scripts. Programming commands control script execution and provide tools such as variables, functions, branching, and loop control.

The Test Script Processor (TSP®) scripting engine is a Lua interpreter. In TSP-enabled instruments, the Lua programming language has been extended with Keithley-specific instrument control commands.

What is Lua?

Lua is a programming language that can be used with TSP-enabled instruments. Lua is an efficient language with simple syntax that is easy to learn.

Lua is also a scripting language, which means that scripts are compiled and run when they are sent to the instrument. You do not compile them before sending them to the instrument.

Lua basics

This section contains the basics about the Lua programming language to allow you to start adding Lua programming commands to your scripts quickly.

For more information about Lua, see the Lua website (*http://www.lua.org*). Another source of useful information is the Lua users group (*http://lua-users.org*), created for and by users of Lua programming language.

Comments

Comments start anywhere outside a string with a double hyphen (--). If the text immediately after a double hyphen (--) is anything other than double left brackets ([[), the comment is a short comment, which continues only until the end of the line. If double left brackets follow the double hyphen (--[[), it is a long comment, which continues until the corresponding double right brackets (]]) close the comment. Long comments may continue for several lines and may contain nested [[...]] pairs. The table below shows how to use code comments.

Using code comments

Type of comment	Comment delimiters	Usage	Example
Short comment	--	Use when the comment text fits on a single line.	`--Turn off the front-panel display.`

Long comment	--[[ ]]	Use when the comment text is longer than one line.	`--[[Display a menu with three menu items.`
			`   If the second menu item is selected,`
			`   the selection will be given the value`
			`   Test2.]]`

Function and variable name restrictions

You cannot use Lua reserved words and top-level command names for function or variable names.

Variable names must contain at least three characters.

The following table lists some of the Lua reserved words. If you attempt to assign these, the event code -285, "TSP Syntax error at line x: unexpected symbol near '*word*' " is displayed, where *word* is the Lua reserved word.

Lua reserved words		
and	for	or
break	function	repeat
do	if	return
else	in	then
elseif	local	true
end	nil	until
false	not	while

Values and variable types

In Lua, you can use variables to store values in the run-time environment for later use.

Lua is a dynamically-typed language; the type of the variable is determined by the value that is assigned to the variable.

Variables in Lua are assumed to be global unless they are explicitly declared to be local. A global variable is accessible by all commands. Global variables do not exist until they have been assigned a value.

Variable types

Variables can be one of the following types.

Variable types and values

Variable type returned	Value	Notes
`"nil"`	not declared	The type of the value `nil`, whose main property is to be different from any other value; usually it represents the absence of a useful value.
`"boolean"`	true or false	Boolean is the type of the values `false` and `true`. In Lua, both `nil` and `false` make a condition `false`; any other value makes it `true`.
`"number"`	number	All numbers are real numbers; there is no distinction between integers and floating-point numbers.
`"string"`	sequence of words or characters	
`"function"`	a block of code	Functions perform a task or compute and return values.
`"table"`	an array	New tables are created with { } braces. For example, {1, 2, 3.00e0}.
`"userdata"`	variables	Allows arbitrary program data to be stored in Lua variables.
`"thread"`	line of execution	

To determine the type of a variable, you can call the `type()` function, as shown in the examples below.

NOTE

The output you get from these examples may vary depending on the data format that is set.

Example: Nil

| ```
x = nil
print(x, type(x))
``` | ```
nil        nil
``` |

Example: Boolean

```y = false``` ```print(y, type(y))```	```false boolean```

Example: Hex constant

You can enter hexadecimal values, but to return a hexadecimal value, you must create a function, as shown in this example. Note that hexadecimal values are handled as a number type.

```
hex = function (i) return "0x"..string.format("%X", i) end
print(hex(0x54|0x55))
print(hex(0x54&0x66))
```

Set the format to return hexadecimal values, then OR two hexadecimal values and AND two hexadecimal values.
Output:
```
0x55
0x44
```

Example: Binary constant

Binary values are returned as floating point decimal values. Note that binary values are handled as a number type.

```x = 0b0000000011111111```
 ```y = 0B1111111100000000```
 ```print(x, type(x))```
 ```print(y, type(y))```
	```255 number```
	 ```65280 number```

Example: String and number

```x = "123"```
 ```print(x, type(x))```
 	
 ```x = x + 7```
 ```print(x, type(x))```
	```123 string```
	
	 Adding a number to x forces its type to number.
	 ```1.30 number```

Example: Function

```function add_two(first_value,```
 ```   second_value)```
 ```   return first_value + second_value```
 ```end```
 ```print(add_two(3, 4), type(add_two))```
	```7 function```

Example: Table

```atable = {1, 2, 3, 4}```
 ```print(atable, type(atable))```
 ```print(atable[1])```
 ```print(atable[4])```
	Defines a table with four numeric elements.
	 Note that the "table" value (shown here as ```a096cd30```) will vary.
	
	 ```table: a096cd30 table```
	 ```1```
	 ```4```

Delete a global variable

To delete a global variable, assign `nil` to the global variable. This removes the global variable from the run-time environment.

Operators

You can compare and manipulate Lua variables and constants using operators.

Arithmetic operators

Operator	Description
+	addition
−	subtraction
*	multiplication
/	division
−	negation (for example, c = −a)
^	exponentiation

Relational operators

Operator	Description
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
~= !=	not equal
==	equal

Bitwise operators

Operator	Description
&	AND
|	OR
^^	exclusive OR
<<	bitwise shift left
>>	bitwise shift right
!	logical NOT

Logical and bitwise operators

The logical operators in Lua are `and`, `or`, and `not`. All logical operators consider both `false` and `nil` as false and anything else as true.

The operator `not` always returns `false` or `true`.

The conjunction operator `and` returns its first argument if the first argument is `false` or `nil`; otherwise, `and` returns its second argument. The disjunction operator `or` returns its first argument if this value is different from `nil` and `false`; otherwise, `or` returns its second argument. Both `and` and `or` use shortcut evaluation, that is, the second operand is evaluated only if necessary.

NOTE

The example output you get may vary depending on the data format settings of the instrument.

Section 7: Introduction to TSP commands   Model DMM7510 7½ Digit Graphical Sampling Multimeter Reference Manual

Example 1

```
print(10 or eventlog.next())        10
print(nil or "a")                   a
print(nil and 10)                   nil
print(false and eventlog.next())    false
print(false and nil)                false
print(false or nil)                 nil
print(10 and 20)                    20
```

Example 2

```
hex = function (i) return "0x"..string.format("%X", i) end
print(hex(0x54 | 0x55))
print(hex(0x54 & 0x66))
```

Set the format to return hexadecimal values, then OR two hexadecimal values and AND two hexadecimal values.
Output:
```
0x55
0x44
```

Example 3

```
hex = function (i) return "0x"..string.format("%X", i) end
a, b= 0b01010100, 0b01100110
print(hex(a), "&", hex(b), "=", hex(a & b))
```

Set the format to return hexadecimal values, define binary values for a and b, then AND a and b.
Output:
```
0x54 & 0x66 = 0x44
```

String concatenation

String operators

Operator	Description
..	Concatenates two strings. If either argument is a number, it is coerced to a string (in a reasonable format) before concatenation.

Example: Concatenation

```
print(2 .. 3)                  23
print("Hello " .. "World")     Hello World
```

7-16   DMM7510-901-01 Rev. A / November 2014

Operator precedence

Operator precedence in Lua follows the order below (from higher to lower priority):

- `^` (exponentiation)
- `not`, `-` (unary), `!` (logical NOT)
- `*`, `/`, `<<`, `>>`
- `+`, `-`, `&`, `|`, `^^`
- `..` (concatenation)
- `<`, `>`, `<=`, `>=`, `~=`, `!=`, `==`
- `and`
- `or`

You can use parentheses to change the precedences in an expression. The concatenation (`".."`) and exponentiation (`"^"`) operators are right associative. All other binary operators are left associative. The examples below show equivalent expressions.

Equivalent expressions

`reading + offset < testValue/2+0.5`	`=`	`(reading + offset) <` `    ((testValue/2)+0.5)`
`3+reading^2*4`	`=`	`3+((reading^2)*4)`
`Rdg < maxRdg and lastRdg <=` `    expectedRdg`	`=`	`(Rdg < maxRdg) and (lastRdg <=` `    expectedRdg)`
`-reading^2`	`=`	`-(reading^2)`
`reading^testAdjustment^2`	`=`	`reading^(testAdjustment^2)`

Functions

With Lua, you can group commands and statements using the `function` keyword. Functions can take zero, one, or multiple parameters, and they return zero, one, or multiple values.

You can use functions to form expressions that calculate and return a value. Functions can also act as statements that execute specific tasks.

Functions are first-class values in Lua. That means that functions can be stored in variables, passed as arguments to other functions, and returned as results. They can also be stored in tables.

Note that when a function is defined, it is stored in the run-time environment. Like all data that is stored in the run-time environment, the function persists until it is removed from the run-time environment, is overwritten, or the instrument is turned off.

Create functions using the function keyword

Functions are created with a message or in Lua code in either of the following forms:

```
function myFunction(parameterX) functionBody end
myFunction = function (parameterX) functionBody end
```

Where:

- `myFunction`: The name of the function.
- `parameterX`: Parameter names. To use multiple parameters, separate the names with commas.
- `functionBody`: The code that is executed when the function is called.

To execute a function, substitute appropriate values for *parameterX* and insert them into a message formatted as:

```
myFunction(valueForParameterX, valueForParameterY)
```

Where *valueForParameterX* and *valueForParameterY* represent the values to be passed to the function call for the given parameters.

NOTE

The output you get from these examples will vary depending on the data format settings of the instrument.

Example 1

`function add_two(first_value,`
`    second_value)`
`    return first_value + second_value`
`end`
`print(add_two(3, 4))`
	Creates a variable named `add_two` that has a variable type of function.
	Output:
	`7`

Example 2

`add_three = function(first_value,`
`    second_value, third_value)`
`    return first_value + second_value +`
`        third_value`
`end`
`print(add_three(3, 4, 5))`
	Creates a variable named `add_three` that has a variable type of function.
	Output:
	`12`

Example 3

`function sum_diff_ratio(first_value,`
`    second_value)`
`    psum = first_value + second_value`
`    pdif = first_value - second_value`
`    prat = first_value / second_value`
`    return psum, pdif, prat`
`end`
`sum, diff, ratio = sum_diff_ratio(2, 3)`
`print(sum)`
`print(diff)`
`print(ratio)`
	Returns multiple parameters (sum, difference, and ratio of the two numbers passed to it).
	Output:
	`5`
	`-1`
	`0.66666666666667`

Create functions using scripts

You can use scripts to define functions. Scripts that define a function are like any other script: They do not cause any action to be performed on the instrument until they are executed. The global variable of the function does not exist until the script that created the function is executed.

A script can consist of one or more functions. Once a script has been run, the computer can call functions that are in the script directly.

For detail on creating functions, see <u>Fundamentals of scripting for TSP</u> (on page 7-4).

Conditional branching

Lua uses the `if`, `else`, `elseif`, `then`, and `end` keywords to do conditional branching.

Note that in Lua, `nil` and `false` are `false` and everything else is `true`. Zero (0) is `true` in Lua.

The syntax of a conditional block is as follows:

```
if expression then
   block
elseif expression then
   block
else
   block
end
```

Where:

- `expression` is Lua code that evaluates to either `true` or `false`
- `block` consists of one or more Lua statements

Example: If

```
if 0 then
   print("Zero is true!")
else
   print("Zero is false.")
end
```

Output:
```
Zero is true!
```

Example: Comparison

```
x = 1
y = 2
if x and y then
   print("Both x and y are true")
end
```

Output:
```
Both x and y are true
```

Example: If and else

```
x = 2
if not x then
   print("This is from the if block")
else
   print("This is from the else block")
end
```

Output:
```
This is from the else
block
```

Example: Else and elseif

```
x = 1
y = 2
if x and y then
   print("'if' expression 2 was not false.")
end

if x or y then
   print("'if' expression 3 was not false.")
end

if not x then
   print("'if' expression 4 was not false.")
else
   print("'if' expression 4 was false.")
end

if x == 10 then
   print("x = 10")
elseif y > 2 then
   print("y > 2")
else
   print("x is not equal to 10, and y is not greater than 2.")
end
```
```
Output:
'if' expression 2 was not false.
'if' expression 3 was not false.
'if' expression 4 was false.
x is not equal to 10, and y is not greater than 2.
```

Loop control

If you need to repeat code execution, you can use the Lua `while`, `repeat`, and `for` control structures. To exit a loop, you can use the `break` keyword.

While loops

To use conditional expressions to determine whether to execute or end a loop, you use `while` loops. These loops are similar to [Conditional branching](#) (on page 7-19) statements.

```
while expression do
   block
end
```

Where:

- *expression* is Lua code that evaluates to either `true` or `false`
- *block* consists of one or more Lua statements

NOTE

The output you get from this example may vary depending on the data format settings of the instrument.

Example: While

```
list = {
   "One", "Two", "Three", "Four", "Five", "Six"}
print("Count list elements on numeric index:")
element = 1
while list[element] do
   print(element, list[element])
   element = element + 1
end
```

This loop exits when `list[element]` = nil.
Output:
```
Count list elements on
   numeric index:
1    One
2    Two
3    Three
4    Four
5    Five
6    Six
```

Repeat until loops

To repeat a command, you use the `repeat ... until` statement. The body of a `repeat` statement always executes at least once. It stops repeating when the conditions of the `until` clause are met.

```
repeat
   block
until expression
```

Where:

- `block` consists of one or more Lua statements

- `expression` is Lua code that evaluates to either `true` or `false`

NOTE

The output you get from this example may vary depending on the data format settings of the instrument.

Example: Repeat until

```
list = {
   "One", "Two", "Three", "Four", "Five", "Six"}
print("Count elements in list using repeat:")
element = 1
repeat
   print(element, list[element])
   element = element + 1
until not list[element]
```

Output:
```
Count elements in list
   using repeat:
1    One
2    Two
3    Three
4    Four
5    Five
6    Six
```

For loops

There are two variations of `for` statements supported in Lua: numeric and generic.

NOTE

In a `for` loop, the loop expressions are evaluated once, before the loop starts.

The output you get from these examples may vary depending on the data format settings of the instrument.

Example: Numeric for

```
list = {"One", "Two", "Three", "Four", "Five", "Six"}
---------- For loop -----------
print("Counting from one to three:")
for element = 1, 3 do
   print(element, list[element])
end
print("Counting from one to four, in steps of two:")
for element = 1, 4, 2 do
   print(element, list[element])
end
```

The numeric `for` loop repeats a block of code while a control variable runs through an arithmetic progression.
Output:
```
Counting from one to three:
1    One
2    Two
3    Three
Counting from one to four, in steps of two:
1    One
3    Three
```

Example: Generic for

```
days = {"Sunday",
   "Monday",      "Tuesday",
   "Wednesday",   "Thursday",
   "Friday",      "Saturday"}

for i, v in ipairs(days) do
   print(days[i], i, v)
end
```

The generic `for` statement works by using functions called iterators. On each iteration, the iterator function is called to produce a new value, stopping when this new value is nil.
Output:
```
Sunday      1    Sunday
Monday      2    Monday
Tuesday     3    Tuesday
Wednesday   4    Wednesday
Thursday    5    Thursday
Friday      6    Friday
Saturday    7    Saturday
```

Break

The `break` statement can be used to terminate the execution of a `while`, `repeat`, or `for` loop, skipping to the next statement after the loop. A break ends the innermost enclosing loop.

Return and break statements can only be written as the last statement of a block. If it is necessary to return or break in the middle of a block, an explicit inner block can be used.

NOTE

The output you get from these examples may vary depending on the data format settings of the instrument.

Example: Break with while statement

``
local numTable = {5, 4, 3, 2, 1}
local k = table.getn(numTable)
local breakValue = 3
while k > 0 do
   if numTable[k] == breakValue then
      print("Going to break and k = ", k)
      break
   end
   k = k - 1
end
if k == 0 then
   print("Break value not found")
end
``
	This example defines a `break` value (`breakValue`) so that the `break` statement is used to exit the `while` loop before the value of k reaches 0.
	Output:
	`Going to break and k = 3`

Example: Break with while statement enclosed by comment delimiters

``
local numTable = {5, 4, 3, 2, 1}
local k = table.getn(numTable)
--local breakValue = 3
while k > 0 do
   if numTable[k] == breakValue then
      print("Going to break and k = ", k)
      break
   end
   k = k - 1
end
if k == 0 then
   print("Break value not found")
end
``
	This example defines a break value (`breakValue`), but the break value line is preceded by comment delimiters so that the break value is not assigned, and the code reaches the value 0 to exit the `while` loop.
	Output:
	`Break value not found`

Example: Break with infinite loop

``a, b = 0, 1
while true do
print(a, b)
a, b = b, a + b
if a > 500 then
break
end
end```	This example uses a `break` statement that causes the while loop to exit if the value of `a` becomes greater than 500. Output: 0 1 1 1 1 2 2 3 3 5 5 8 8 13 13 21 21 34 34 55 55 89 89 144 144 233 233 377 377 610

Tables and arrays

Lua makes extensive use of the data type table, which is a flexible array-like data type. Table indices start with 1. Tables can be indexed not only with numbers, but with any value except `nil`. Tables can be heterogeneous, which means that they can contain values of all types except `nil`.

Tables are the sole data structuring mechanism in Lua. They may be used to represent ordinary arrays, symbol tables, sets, records, graphs, trees, and so on. To represent records, Lua uses the field `name` as an index. The language supports this representation by providing `a.name` as an easier way to express `a["name"]`.

NOTE

The output you get from this example may vary depending on the data format settings of the instrument.

Example: Loop array

``atable = {1, 2, 3, 4}
i = 1
while atable[i] do
print(atable[i])
i = i + 1
end```	Defines a table with four numeric elements. Loops through the array and prints each element. The Boolean value of `atable[index]` evaluates to `true` if there is an element at that index. If there is no element at that index, `nil` is returned (`nil` is considered to be `false`). **Output:** 1 2 3 4

Standard libraries

In addition to the standard programming constructs described in this document, Lua includes standard libraries that contain useful functions for string manipulation, mathematics, and related functions. Test Script Processor (TSP®) scripting engine instruments also include instrument control extension libraries, which provide programming interfaces to the instrumentation that can be accessed by the TSP scripting engine. These libraries are automatically loaded when the TSP scripting engine starts and do not need to be managed by the programmer.

The following topics provide information on some of the basic Lua standard libraries. For additional information, see the Lua website (*http://www.lua.org*).

NOTE
When referring to the Lua website, please be aware that the TSP scripting engine uses Lua 5.0.2.

Base library functions

Base library functions

Function	Description
`collectgarbage()` `collectgarbage(limit)`	Sets the garbage-collection threshold to the given limit (in kilobytes) and checks it against the byte counter. If the new threshold is smaller than the byte counter, Lua immediately runs the garbage collector. If there is no limit parameter, it defaults to zero (0), which forces a garbage-collection cycle. See the "Lua memory management" topic below for more information.
`gcinfo()`	Returns the number of kilobytes of dynamic memory that the Test Script Processor (TSP®) scripting engine is using, and returns the present garbage collector threshold (also in kilobytes). See the "Lua memory management" topic below for more information.
`tonumber(x)` `tonumber(x, base)`	Returns `x` converted to a number. If `x` is already a number, or a convertible string, the number is returned; otherwise, it returns `nil`. An optional argument specifies the base to use when interpreting the numeral. The base may be any integer from 2 to 36, inclusive. In bases above 10, the letter `A` (in either upper or lower case) represents 10, `B` represents 11, and so forth, with `Z` representing 35. In base 10, the default, the number may have a decimal part, as well as an optional exponent. In other bases, only unsigned integers are accepted.
`tostring(x)`	Receives an argument of any type and converts it to a string in a reasonable format.
`type(v)`	Returns (as a string) the type of its only argument. The possible results of this function are `"nil"` (a string, not the value `nil`), `"number"`, `"string"`, `"boolean"`, `"table"`, `"function"`, `"thread"`, and `"userdata"`.

Lua memory management

Lua automatically manages memory, which means you do not have to allocate memory for new objects and free it when the objects are no longer needed. Lua occasionally runs a garbage collector to collect all objects that are no longer accessible from Lua. All objects in Lua are subject to automatic management, including tables, variables, functions, threads, and strings.

Lua uses two numbers to control its garbage-collection cycles. One number counts how many bytes of dynamic memory Lua is using; the other is a threshold. When the number of bytes crosses the threshold, Lua runs the garbage collector, which reclaims the memory of all inaccessible objects. The byte counter is adjusted and the threshold is reset to twice the new value of the byte counter.

String library functions

This library provides generic functions for string manipulation, such as finding and extracting substrings. When indexing a string in Lua, the first character is at position 1 (not 0, as in ANSI C). Indices may be negative and are interpreted as indexing backward from the end of the string. Thus, the last character is at position −1, and so on.

String library functions

Function	Description

`string.byte(s)`
`string.byte(s, i)`
`string.byte(s, i, j)`
	Returns the internal numeric codes of the characters $s[i]$, $s[i+1]$, $\cdots$, $s[j]$. The default value for i is 1; the default value for j is i.

`string.char(···)`	Receives zero or more integers separated by commas. Returns a string with length equal to the number of arguments, in which each character has the internal numeric code equal to its corresponding argument.

`string.format(`
   `formatstring, ...)`
	Returns a formatted version of its variable number of arguments following the description given in its first argument, which must be a string. The format string follows the same rules as the `printf` family of standard C functions. The only differences are that the modifiers `*`, `l`, `L`, `n`, `p`, and `h` are not supported and there is an extra option, `q`. The `q` option formats a string in a form suitable to be safely read back by the Lua interpreter; the string is written between double quotes, and all double quotes, newlines, embedded zeros, and backslashes in the string are correctly escaped when written.
	For example, the call:
	`string.format('%q', 'a string with "quotes" and`
	   `\n new line')`
	will produce the string:
	`"a string with \"quotes\" and \`
	   `new line"`
	The options `c`, `d`, `E`, `e`, `f`, `g`, `G`, `i`, `o`, `u`, `X`, and `x` all expect a number as argument. `q` and `s` expect a string. This function does not accept string values containing embedded zeros, except as arguments to the `q` option.

`string.len(s)`	Receives a string and returns its length. The empty string `""` has length `0`. Embedded zeros are counted, so `"a\000bc\000"` has length `5`.

String library functions

Function	Description
string.lower(*s*)	Receives a string and returns a copy of this string with all uppercase letters changed to lowercase. All other characters are left unchanged.
string.rep(*s*, *n*)	Returns a string that is the concatenation of *n* copies of the string *s*.
string.sub(*s*, *i*) string.sub(*s*, *i*, *j*)	Returns the substring of *s* that starts at *i* and continues until *j*; *i* and *j* can be negative. If *j* is absent, it is assumed to be equal to -1 (which is the same as the string length). In particular, the call string.sub(*s*, 1, *j*) returns a prefix of *s* with length *j*, and string.sub(*s*, -*i*) returns a suffix of *s* with length *i*.
string.upper(*s*)	Receives a string and returns a copy of this string with all lowercase letters changed to uppercase. All other characters are left unchanged.

Math library functions

This library is an interface to most of the functions of the ANSI C math library. All trigonometric functions work in radians. The functions `math.deg()` and `math.rad()` convert between radians and degrees.

Math library functions

Function	Description
`math.abs(x)`	Returns the absolute value of x.
`math.acos(x)`	Returns the arc cosine of x.
`math.asin(x)`	Returns the arc sine of x.
`math.atan(x)`	Returns the arc tangent of x.
`math.atan2(y, x)`	Returns the arc tangent of y/x, but uses the signs of both parameters to find the quadrant of the result (it also handles correctly the case of x being zero).
`math.ceil(x)`	Returns the smallest integer larger than or equal to x.
`math.cos(x)`	Returns the cosine of x.
`math.deg(x)`	Returns the angle x (given in radians) in degrees.
`math.exp(x)`	Returns the value e^x.
`math.floor(x)`	Returns the largest integer smaller than or equal to x.
`math.frexp(x)`	Returns m and e such that $x = m2^e$, where e is an integer and the absolute value of m is in the range $[0.5, 1]$ (or zero when x is zero).
`math.ldexp(m, e)`	Returns $m2^e$ (e should be an integer).
`math.log(x)`	Returns the natural logarithm of x.
`math.log10(x)`	Returns the base-10 logarithm of x.
`math.max(x, ...)`	Returns the maximum value among its arguments.
`math.min(x, ...)`	Returns the minimum value among its arguments.
`math.pi`	The value of π (3.141592654).
`math.pow(x, y)`	Returns x^y (you can also use the expression $x\char94 y$ to compute this value).
`math.rad(x)`	Returns the angle x (given in degrees) in radians.

`math.random()`
`math.random(m)`
`math.random(m, n)`
	This function is an interface to the simple pseudorandom generator function `rand` provided by ANSI C.
	When called without arguments, returns a uniform pseudorandom real number in the range $[0,1]$. When called with an integer number m, `math.random()` returns a uniform pseudorandom integer in the range $[1, m]$. When called with two integer numbers m and n, `math.random()` returns a uniform pseudorandom integer in the range $[m, n]$.

`math.randomseed(x)`	Sets x as the seed for the pseudorandom generator: equal seeds produce equal sequences of numbers.
`math.sin(x)`	Returns the sine of x.
`math.sqrt(x)`	Returns the square root of x. (You can also use the expression $x\char94 0.5$ to compute this value.)
`math.tan(x)`	Returns the tangent of x.

Test Script Builder (TSB)

Keithley Instruments Test Script Builder (TSB) is a software tool available from the Keithley Instruments website. You can install and use TSB to develop scripts for TSP-enabled instruments.

You must use the TSP command set to use Test Script Builder. Refer to Determining the command set you will use (on page 2-86) for information about the command sets and changing them.

Installing the TSB software

The installation files for the Test Script Builder software are available for download on the Keithley Instruments support website (*http://www.keithley.com/support*).

To install the Test Script Builder (TSB) software:

1. Close all programs.
2. Download the installer to your computer and double-click the `.exe` file to start the installation.
3. Follow the on-screen instructions.

Installing the TSB add-in

When you install the Test Script Builder Software Suite, all available updates for TSB Add-in software are also installed. This includes any additional tools for the Test Script Builder (TSB) and model-specific examples and help files (see Installing the TSB software (on page 7-29)). If you have an existing version of TSB that does not have model-specific examples for an instrument you are using, you can download a separate add-in from the Keithley Instruments support website (*http://www.keithley.com/support*).

Before installing the TSB Add-in software, you must install the TSB software.

To install the TSB Add-in software:

1. Close all programs.
2. Download the Add-in to your computer and double-click it to start installation.
3. Follow the on-screen instructions.

Using Test Script Builder (TSB)

Keithley Instruments Test Script Builder (TSB) is a software tool that simplifies building test scripts. You can use TSB to perform the following operations:

- Send remote commands and Lua statements
- Receive responses (data) from commands and scripts
- Upgrade instrument firmware
- Create, manage, and run user scripts
- Debug scripts
- Import factory scripts to view or edit and convert to user scripts

The Keithley Instruments Test Script Processor (TSP$^®$) scripting engine is a Lua interpreter. In TSP-enabled instruments, the Lua programming language has been extended with Keithley-specific instrument control commands. For more information about using the Lua scripting language with Keithley TSP-enabled instruments, refer to the <u>Fundamentals of programming for TSP</u> (on page 7-11) section.

Keithley has created a collection of remote commands specifically for use with Keithley TSP-enabled instruments; for detailed information about those commands, refer to the "Command reference" section of the documentation for your specific instrument. You can build scripts from a combination of these commands and Lua programming statements. Scripts that you create are referred to as "user scripts." Also, some TSP-enabled instruments come with a number of built-in factory scripts.

The following figure shows an example of the Test Script Builder. As shown, the workspace is divided into these areas:

- Project navigator
- Script editor
- Outline view
- Programming interaction
- Help files

Figure 146: Example of the Test Script Builder workspace

Item	Description
1	Project navigator
2	Script editor; right-click to run the script that is displayed
3	Outline view
4	Programming interaction
5	Help; includes detailed information on using Test Script Builder

Project navigator

The project navigator consists of project folders and the script files (.tsp) created for each project. Each project folder can have one or more script files.

To view the script files in a project folder, click the plus (+) next to the project folder. To hide the folder contents, click the minus (−) next to the project folder.

You can download a TSP project to the instrument and run it, or you can run it from the TSB interface.

Script editor

The script editor is where you write, modify, and debug scripts.

To open and display a script file, double-click the file name in the project navigator. You can have multiple script files open in the script editor at the same time. Each open script file is displayed on a separate tab.

To display another script file that is already open, click the tab that contains the script in the script editor area.

Outline view

The outline view allows you to navigate through the structure of the active script in the script editor. Double-clicking a variable name or icon causes the first instance of the variable in the active script to be highlighted.

This view shows:

- Names of local and global variables
- Functions referenced by the active script in the script editor
- Parameters
- Loop control variables
- Table variables
- Simple assignments to table fields

Programming interaction

This part of the workspace is where you interact with the scripts that you are building in Test Script Builder (TSB). The actual contents of the programming interaction area of the workspace can vary.

You can send commands from the Instrument Console command line, retrieve data, view variables and errors, and view and set breakpoints when using the debug feature.

Connecting an instrument in TSB

You must use the TSP command set with the Test Script Builder software. For information on changing the command set, refer to Determining the command set you will use (on page 2-86).

To connect the Test Script Builder software to an instrument:

1. Click the **Open Instrument** icon in the script editor toolbar.

Figure 147: Opening an instrument connection in TSB

2. The Select Instrument dialog box opens. Select an existing instrument from the list, or type the VISA resource ID of the instrument in the **Select or enter new instrument** box.
3. If needed, enter a password.

Figure 148: Select Instrument dialog box

4. Click **OK**. You briefly see the Opening Resource dialog box, and then the instrument is visible in the Instrument Console.

Figure 149: Instrument connected in TSB

Instrument open in Test Script Builder

Creating a new TSP project

To create a new Test Script Processor (TSP®) project:

1. On the **File** menu in the TSP perspective, select **New > TSP Project**. The New TSP Project dialog box opens.

Figure 150: New TSP Project dialog box

2. Type a name for your project in the **Project name** box.
3. Select the location to create the new project.
4. Click **Finish**. The new project appears in the list of projects in the project navigator, and a file named `main.tsp` is created in the project. You can rename the `.tsp` file.
5. If you do not want to build your project automatically when it is saved or run, from the **Project** menu, clear **Build Automatically**.

NOTE

If you make changes to your project and do not build it before you run it, the Problems tab may not appear when problems are encountered.

Adding a new TSP file to a project

To add a new TSP file to a project:

1. Select the **File** menu and select **New > TSP File**. The New TSP File dialog box opens.
2. Select the project folder where you want to save the file.
3. Enter a name in the **File name** box.
4. Click **Finish**.

Running a script

You can run a script in the Test Script Builder (TSB) software using any of the following methods:

*   Run a script that is open in the script editor area
*   Run scripts that are listed in the Navigator area that are not currently open in the script editor window
*   Run a collection of scripts by creating a run configuration (see <u>Creating a run configuration</u> (on page 7-36))

<table>
<tr><td align="center">NOTE</td></tr>
</table>

When you use any of the run controls to run a script, the area that has focus in the workspace is important. For example, if the Navigator area is active (the tab is shaded) when you click the **Run** icon, the script file that is highlighted in the Navigator area is run instead of the active script in the script editor area.

The following list describes the most commonly used controls to run scripts in TSB:

*   Right-click in the script editor area and select **Run Editor Contents** to run the active script as it currently appears in the script editor
*   Right-click in the script editor area and select **Run As > 1 TSP File** to run the last saved version of the active script in the script editor as a `.tsp` file
*   Select an action from the **Run** menu at the top of the TSB software interface

Creating a run configuration

A run configuration allows you to download multiple script files to an instrument and execute them as a single script.

To create a run configuration:

1. On the **Run** menu, select **Run Configurations**. The Run Configurations dialog box opens.

2. The left pane of the dialog box lists existing run and debug configurations. Select the script where the Run Configuration will be saved.

3. Click the **New launch configuration** icon ⬜ at the top left of the dialog box. By default, a new configuration is created with the name `New_configuration`.

Figure 151: Run Configurations dialog box

4. In the **Name** box, enter the name of your new run configuration.
5. Click the **Browse** button next the Project box.
6. Select a project from the list of available projects
7. Click **OK**.

Figure 152: Project Selection dialog box

The TSP files for the selected project are added to the Available Project Files list on the Main tab.

Figure 153: Available files for selected project

8. Select the files you want to add to the run configuration and click **Add** to add them to the Load Order list.

   To change the load order of the TSP files, select the files you want to move and click **Up** or **Down** until the files are in the correct order.

Figure 154: Selected TSP files load order

9. Click **Apply**.
10. Click the **Script Attributes** tab.
11. Select one of the following:

    Load and Execute: If you select this option, which is the default selection, the script automatically loads into the instrument's volatile memory (run-time environment) and executes when you click **Run**.

    Load: If you select this option, the script is loaded into the instrument's volatile memory when you click **Run**, but is not executed until you manually run it. To manually run it from the command line in the Instrument Console, type *MyConfiguration*.run() (where *MyConfiguration* is the name of your configuration).

Figure 155: Script Attributes tab

12. In the Storage area of the Script Attributes tab, select **Volatile** or **Non-volatile**. For products that support autorun scripts, if you select Non-volatile, you can select **Auto Run** to have the script run automatically when the instrument is turned on.
    Note that all scripts are initially stored in the volatile (runtime) memory of the instrument memory and are lost if you turn the instrument power off and then on again. If you want to keep the script on the instrument through a power cycle, select **Non-volatile** storage.
13. In the Debug area of the Script Attributes tab, you can select **Generate Debug File**.
    When you select this option, a Debug subfolder is created in your test folder, and a file with a .DBG extension is created in that folder. Note that this is a feature of the Eclipse platform, and you will not use this file to debug your script. It contains all the scripts in your run configuration, so that you can see them together in the order in which they will load.
14. Click **Close** or **Run**. The run configuration is added to the run configurations list.

NOTE

To run the last used run configuration, click the **Run** icon in the main TSB toolbar. To run a different run configuration, right-click in the script editor area and select **Run As > Run Configurations**. Select a different run configuration, and then click **Run** in the Run Configurations dialog box.

Memory considerations for the run-time environment

The Model DMM7510 reserves a large amount of memory for use with interactions with the front panel, SCPI and commands, and test scripts. The amount of memory usage is affected by the following product features:

- Reading buffers (including local default and user-created reading buffers; if they are on a remote node, they only affect the remote node)
- Lua variables (large arrays)
- TriggerFlow trigger models
- Configuration lists

The more a feature is used or the larger its definition, the more memory it consumes. For normal usage, reading buffers commonly reserve large amounts of memory. The amount of memory used depends on the number of readings and the buffer style.

> ### ⚠ CAUTION
>
> The Model DMM7510 notifies you when the system runs out of memory. If the instrument encounters memory allocation errors (errors that specifically state "Out of Memory"), the state of the instrument cannot be guaranteed. After attempting to save any important data, turn off power to the instrument and turn it back on to reset the runtime environment and return the instrument to a known state. Unsaved scripts and reading buffers will be lost.

If you encounter memory problems, examine the test script or SCPI commands that were being executed when the memory problems occurred. Take action to reduce the size of the elements that are consuming memory. If you are using TSP commands and scripting, also consider using the `collectgarbage()` command to clean up unused memory. For information on `collectgarbage()`, refer to Base library functions (on page 7-25).

The default size settings for the default reading buffers (`defbuffer1` and `defbuffer2`) are large. If your application does not use these buffers, you can set them to the minimum of 10 readings to conserve space. For information on adjusting the buffer size, refer to Setting reading buffer capacity (on page 3-22).

The buffer style is set when you create a user-defined reading buffer. The buffer style cannot be changed, so to eliminate memory problems caused by the style, you may need to delete or adjust the capacity of the buffers. Refer to Creating buffers (on page 3-14) for information on the effects of styles.

TSP command reference

In this section:

TSP command programming notes

This section contains general information about using TSP commands.

TSP syntax rules

This section provides rules for what you can and cannot do when entering TSP commands.

Upper and lower case

Instrument commands are case sensitive.

Function and attribute names are in lowercase characters.

Parameters and attribute constants can use a combination of lowercase and uppercase characters. The correct case for a specific command is shown in its command description.

The following example shows the `beeper.beep()` function, where `2` is the duration in seconds and `2400` is the frequency. Note that the function is in lowercase characters:

```
beeper.beep(2, 2400)
```

The following command changes the display light state to be at level 50. Note that the attribute (`display.lightstate`) is lower case, but the constant (`display.STATE_LCD_50`) is a combination of lowercase and uppercase characters:

```
display.lightstate = display.STATE_LCD_50
```

White space

You can send commands with or without white spaces.

For example, the following functions, which set the length and frequency of the instrument beeper, are equivalent:

```
beeper.beep(2,2400)
beeper.beep (2,  2400)
```

Parameters for functions

All functions must have a set of parentheses `()` immediately following the function. If there are parameters for the function, they are placed between the parentheses. The parentheses are required even when no parameters are specified.

The following example shows the `beeper.beep()` function, where `2` is the duration in seconds and `2400` is the frequency. Note that the parameters are inside the parentheses:

```
beeper.beep(2, 2400)
```

The command below resets commands to their default values (no parameters are needed):

```
reset()
```

Multiple parameters

Multiple parameters must be separated by commas.

For example, the following commands set the beeper to emit a double-beep at 2400 Hz, with a beep sequence of 0.5 s on, a delay of 0.25 s, and then 0.5 s on:

```
beeper.beep(0.5, 2400)
delay(0.250)
beeper.beep(0.5, 2400)
```

Time and date values

Time and date values are represented as the number of seconds since some base. The time bases are:

- **UTC 12:00 am Jan 1, 1970:** Some examples of UTC time are reading buffer timestamps, calibration adjustment and verification dates, and the value returned by `os.time()`.
- **Event:** Time referenced to an event, such as the first reading stored in a reading buffer.

Local and remote control

The instrument can be controlled locally or remotely.

When the instrument is controlled locally, you operate the instrument using the front-panel controls. When it is controlled remotely, you operate the instrument through a controller (usually a computer). When the instrument is first powered on, it is controlled locally.

The type of control is displayed on the front panel. When the instrument is in local control, the REMOTE LED indicator in the upper right corner is off and the control indicator on the upper left of the screen shows Local.

When the instrument is in remote control, the front-panel REMOTE LED indicator is on and the control indicator at the top left of the screen shows the type of communication interface.

Remote control

When the instrument is controlled remotely, the front-panel controls are disabled. You can still view information on the front-panel display and move between the screens using the keys and touchscreen controls. If you change a selection, however, you are prompted to switch control to local.

To switch to remote control:

* Send a command from the computer to the instrument.
* Open communications between the instrument and Test Script Builder.

Local control

To change to local control, you can:

* Choose an option from the screens and try to change the value; select **Yes** on the dialog box that is displayed.
* Send the logout command from the computer.
* Turn the instrument off and on.

Using the TSP command reference

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

Figure 156: Example instrument command description

feature.enable

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	feature.OFF

Usage

```
state = feature.enable
feature.enable = state
```

state	Disable the example feature: feature.OFF
	Enable the example feature: feature.ON

Details

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

Example

feature.enable = feature.ON	Enables the example feature.

Also see

exampleUnit.enable (on page 1-73)

The subsections contain information about the command. The subsections are:

- Command name, brief description, and summary table
- Usage
- Details
- Example
- Also see

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

Command name, brief description, and summary table

Each instrument command description starts with the command name, followed by a brief description and a table with information for each command. Descriptions of the numbered items in the figure below are provided below.

Figure 157: TSP command name and summary table

feature.enable

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	feature.OFF

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

2 **Type of command**. Commands can be functions, attributes, or constants. If the command is an attribute, it can be read-only (R), read-write (RW), or write-only (W).

3 **TSP-Link accessible**. Indicates whether or not the command can be accessed through a TSP-Link network (Yes or No).

4 **Affected by**. This columns lists commands or actions that can change the value of the command, including.

- **Power cycle:** The command settings are not saved through a power cycle.
- **Restore configuration:** If you restore a configuration script, this setting changes to the stored setting.
- **Instrument reset:** When you reset the instrument, this command is reset to its default value. Reset can be done from the front panel or when you send `reset()` or `*RST`.
- **Measure configuration list:** If you recall a measure configuration list, this setting changes to the setting stored in the list.
- **Function:** This command changes value when the function is changed (for example, changing from a voltage measurement function to a current measurement function.

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

- **Not saved:** Command is not saved and must be typed each time you use it.
- **Nonvolatile memory:** The command is stored in a storage area in the instrument where information is saved even when the instrument is turned off.
- **Configuration script:** Command is saved as part of the configuration script.
- **Measure configuration list:** This command is stored in measure configuration lists.

6   **Default value:** Lists the default value or constant for the command.

Command usage

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

Figure 158: TSP usage description

1   **Structure of command usage:** Shows how the parts of the command should be organized. If a parameter is shown to the left of the command, it is the return when you print the command. Information to the right are the parameters or other items you need to enter when setting the command.

2   **User-supplied parameters:** Indicated by italics. For example, for the function `beeper.beep(`*`duration, frequency`*`)`, replace *`duration`* with the number of seconds and *`frequency`* with the frequency of the tone. `beeper.beep(2, 2400)` generates a two-second, 2400 Hz tone.

Some commands have optional parameters. If there are optional parameters, they must be entered in the order presented in the Usage section. You cannot leave out any parameters that precede the optional parameter. Optional parameters are shown as separate lines in usage, presented in the required order with each valid permutation of the optional parameters. For example:
`printbuffer(`*`startIndex`*`, `*`endIndex`*`, `*`buffer1`*`)`
`printbuffer(`*`startIndex`*`, `*`endIndex`*`, `*`buffer1`*`, `*`buffer2`*`)`

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

Command details

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

Figure 159: TSP Details description

Example section

The Example section of the remote command description shows examples of how you can use the command.

Figure 160: TSP example code

1 Actual example code that you can copy from this table and paste into your own programming application.

2 Description of the code and what it does. This may also contain example output of the code.

Related commands and information

The Also See section of the remote command description lists additional commands or sections that are related to the command.

Figure 161: TSP Also See description

TSP commands

acal.count

This attribute returns the number of times automatic calibration has been run.

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

Usage

```
value = acal.count
```

value	The number of times auto calibration has been run

Details

The number of times that auto calibration has been run since the last factory calibration. The count restarts at 1 after a factory calibration.

Example

print(acal.count)	Returns the number of times auto calibration has been run. Example output: 15

Also see

Auto calibration (on page 3-40)
acal.run() (on page 8-12)

acal.lastrun.internaltemp

This attribute returns the internal temperature of the instrument when auto calibration was run.

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

Usage

temperature = acal.lastrun.internaltemp

temperature	The internal temperature

Details

The temperature is displayed in Celsius (°C).

The instrument updates the internal temperature value when the instrument refreshes autozero. If autozero is set to off or if autozero is not available for the selected function (such as capacitance, continuity, frequency or period), the internal temperature value is not updated.

Example

print(acal.lastrun.internaltemp)	Returns the internal temperature of the instrument when auto calibration was last run. Example output: 63.167084

Also see

acal.lastrun.tempdiff (on page 8-9)
acal.run() (on page 8-12)
acal.lastrun.tempdiff (on page 8-9)

acal.lastrun.tempdiff

This attribute returns the difference between the internal temperature and the temperature when auto calibration was last run.

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

Usage

```
temperature = acal.lastrun.tempdiff
```

temperature	The internal temperature

Details

The temperature is displayed in Celsius (°C).

The instrument updates the internal temperature value when the instrument refreshes autozero. If autozero is set to off or if autozero is not available for the selected function (such as capacitance, continuity, frequency or period), the internal temperature value is not updated.

Example

`print(acal.lastrun.tempdiff)`	Returns the difference between the temperature of the instrument when auto calibration was run and the present internal temperature. Example output: `4.5678`

Also see

acal.lastrun.internaltemp (on page 8-8)
acal.run() (on page 8-12)

acal.lastrun.time

This attribute returns the date and time when auto calibration was last run.

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

Usage

```
dateTime = acal.lastrun.time
```

dateTime	The date and time

Details

The date and time is returned in the format:

```
MM/DD/YYYY HH:MM:SS.NNNNNNNNN
```

Where:

- `MM/DD/YYYY` is the month, date, and year

- `HH:MM:SS.NNNNNNNNN` is the hour, minute, second, and fractional second

Example

`print(acal.lastrun.time)`	Returns the date and time when the auto calibration was last run. Example output: `05/28/2014 00:37:47.743000000`

Also see

[acal.run()](#) (on page 8-12)
[Auto calibration](#) (on page 3-40)

acal.nextrun.time

This attribute returns the date and time when the next auto calibration is scheduled to be run.

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

Usage

```
dateTime = acal.nextrun.time
```

dateTime	The date and time when auto calibration is scheduled to be run

Details

The date and time is returned in the format:

`MM/DD/YYYY HH:MM:SS.NNNNNNNNN`

Where:

- `MM/DD/YYYY` is the month, date, and year

- `HH:MM:SS.NNNNNNNNN` is the hour, minute, second, and fractional second

Example

`print(acal.nextrun.time)`	Returns date and time when auto calibration is next scheduled to be run. Example output: `05/29/2014 17:11:17.000000000`

Also see

Auto calibration (on page 3-40)
acal.run() (on page 8-12)
acal.schedule() (on page 8-13)

acal.revert()

This function returns auto calibration constants to the previous constants.

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

Usage

```
acal.revert()
```

Details

This command reverts the present set of auto calibration constants to the previous set of auto calibration constants.
The last run time and internal temperature are reverted to the previous values. The auto calibration count is not changed.

Example

`acal.revert()`	Auto calibration values are reverted to the previous set of auto calibration constants.

Also see

acal.run() (on page 8-12)

acal.run()

This function immediately runs auto calibration and stores the constants.

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

Usage

```
acal.run()
```

Details

During auto calibration, a progress message is displayed on the front panel. At completion, an event message is generated.
If you have set up auto calibration to run at a scheduled interval, when you send the run command, the instrument adjusts the next scheduled auto calibration to be the next interval. For example, if auto calibration is scheduled to run every 7 days, but you run auto calibration on day 3, the next auto calibration will run 7 days after day 3.

Example

`acal.run()`	Auto calibration starts running. When it is complete, an information message is generated in the event log.

Also see

acal.schedule() (on page 8-13)
Auto calibration (on page 3-40)
localnode.internaltemp (on page 8-218)

acal.schedule()

This function sets how often auto calibration occurs or prompts you to run it.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration	Nonvolatile memory Configuration script	Run every 8 hours starting at midnight

Usage

```
action, interval, hour = acal.schedule()
acal.schedule(acal.ACTION_NONE)
acal.schedule(action, interval)
acal.schedule(action, interval, hour)
```

action	Determines when and if the instrument automatically runs auto calibration: • To run auto calibration at the scheduled time: `acal.ACTION_RUN` • To notify you that the auto calibration needs to be run at the scheduled time: `acal.ACTION_NOTIFY` • To turn off scheduling: `acal.ACTION_NONE`; no other parameters are needed if none is selected
interval	Determines how often auto calibration should be run or notification should occur: • Every 8 hours: `acal.INTERVAL_8HR` • Every 16 hours: `acal.INTERVAL_16HR` • Every day: `acal.INTERVAL_1DAY` • Every 7 days: `acal.INTERVAL_7DAY` • Every 14 days: `acal.INTERVAL_14DAY` • Every 30 days: `acal.INTERVAL_30DAY` • Every 90 days: `acal.INTERVAL_90DAY`
hour	Specify when the auto calibration should occur; specify in 24-hour time format (`0` to `23`; default is `0`); not available for the 8-hour or 16-hour interval

Details

Auto calibration does not start until actions that are active on the instrument are complete. For example, if a trigger model is running when auto calibration is scheduled to run, auto calibration does not start until the trigger model stops.

If the scheduled time for auto calibration occurs before the warmup period completes, the instrument will not start auto calibration. The instrument waits until the warmup period is complete before starting a scheduled auto calibration. A message is displayed when warmup is complete and auto calibration is going to run.

If the instrument is powered off when an auto calibration was scheduled, auto calibration is run as soon as the warmup period is complete when the instrument is powered on.

You can run auto calibration manually even if a scheduled auto calibration is set.

When auto calibration is scheduled to run at a scheduled interval, but it runs at a time other than the scheduled interval, subsequent scheduled intervals are adjusted according to the actual auto calibration start time.

Example

```
acal.schedule(acal.ACTION_RUN, acal.INTERVAL_1DAY, 8)
```

Sets auto calibration to run every day at 8 am.

Also see

Auto calibration (on page 3-40)
acal.run() (on page 8-12)

beeper.beep()

This function generates an audible tone.

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

Usage

```
beeper.beep(duration, frequency)
```

duration	The amount of time to play the tone (0.001 to 100 s)
frequency	The frequency of the beep (20 to 8000 Hz)

Details

You can use the beeper of the instrument to provide an audible signal at a specific frequency and time duration.

Example

`beeper.beep(2, 2400)`	Generates a 2 s, 2400 Hz tone.

Also see

None

buffer.clearstats()

This function clears the statistical information associated with the specified buffer.

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

Usage

```
buffer.clearstats()
buffer.clearstats(bufferVar)
```

bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; defaults to `defbuffer1` if not specified

Details

This command clears the statistics without clearing the readings.

Example

`buffer.clearstats()`	Clears statistics for `defbuffer1`.
`buffer.clearstats(testData)`	Clears statistics for `testData`.

Also see

buffer.getstats() (on page 8-16)

buffer.delete()

This function deletes a user-defined reading buffer.

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

Usage

```
buffer.delete(bufferName)
```

bufferName	The name of a user-defined reading buffer

Details

You cannot delete the default reading buffers, defbuffer1 and defbuffer2.

Example

```
buf400 = buffer.make(400)
dmm.measure.read(buf400)
printbuffer(1, buf400.n, buf400.relativetimestamps)
buffer.delete(buf400)
```

Create a 400-element reading buffer named buf400.

Make measurements and store the readings in buf400.

Print the relative timestamps for each reading in the buffer.

Example output, assuming five readings are stored in the buffer:

0, 0.412850017, 0.821640085, 1.230558058, 1.629523236

Delete buf400.

Also see

buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

buffer.getstats()

This function returns statistics from a specified reading buffer.

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

Usage

```
buffer.getstats()
buffer.getstats(bufferVar)
statsVar = buffer.getstats(bufferVar)
```

bufferVar	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined buffer; if no buffer is specified, this parameter defaults to defbuffer1
statsVar	A table with the following entries: n, min, max, mean, and stddev; see **Details** for information on the entries

Details

This function returns a table with statistical data about the data that was placed in the reading buffer.

The instrument automatically updates reading buffer statistics as data is added to the reading buffer. When the reading buffer is configured to fill continuously and overwrite older data with new data, the buffer statistics include the data that was overwritten.

The table returned from this function provides statistics at the time the function is called. Although the instrument continues to update the statistics, the table that is returned is not updated. To get fresh statistics, call this function again.

The `statsVar` parameter contains the values described in the following table.

Attribute	When returned	Description
min	n > 0	A table that contains data about the minimum reading value that was added to the buffer
mean	n > 0	The average of all readings added to the buffer
stddev	n > 1	The standard deviation of all readings that were added to the buffer
n	Always	The number of data points on which the statistics are based
max	n > 0	A table that contains data about the maximum reading value that was added to the buffer

If `n` equals zero (0), all other values are `nil`. If `n` equals 1, `stddev` is `nil` because the standard deviation of a sample size of 1 is undefined.

Use the following command to get `statsVar`; a table with the following entries in it: n, min, max, mean, and stddev.

```
statsVar = buffer.getstats(bufferVar)
```

Use the following commands to print these entries:

```
print(statsVar.n)
print(statsVar.mean)
print(statsVar.stddev)
print(statsVar.min.reading)
print(statsVar.min.timestamp)
print(statsVar.min.seconds)
print(statsVar.min.fractionalseconds)
print(statsVar.max.reading)
print(statsVar.max.seconds)
print(statsVar.max.fractionalseconds)
print(statsVar.max.timestamp)
```

The commands that return minimum and maximum values each also return tables. These tables contain the following values:

Attribute	Description
reading	The reading value
timestamp	The `min.timestamp` is the timestamp of the minimum data point in the buffer and the `max.timestamp` is the timestamp of the maximum data point in the buffer

Example

`print(buffer.getstats(defbuffer1))`	Get statistics on `defbuffer1`. Returns a table. Output: `17ac630` The table has the following entries in it: `n, min, max, mean, stddev`
`defBufStats = buffer.getstats(defbuffer1)` `print(defBufStats)`	Assign the name `defBufStats` to the table. Get statistics for the default reading buffer named `defbuffer1` Returns the `defBufStats` table with the following entries in it: `n, min, max, mean, stddev`

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

buffer.make()

This function creates a user-defined reading buffer.

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

Usage

```
bufferVar = buffer.make(bufferSize)
bufferVar = buffer.make(bufferSize, style)
```

`bufferVar`	A user-supplied string that indicates the name of the buffer
`bufferSize`	The maximum number of readings that can be stored in `bufferVar`; minimum is `10`
`style`	The type of reading buffer to create: • Store readings with reduced accuracy (6.5 digits) with no formatting information, 1 μs accurate timestamp, maximum 27,500,000 readings: `buffer.STYLE_COMPACT` • Store readings with full accuracy with formatting, maximum 11,000,000 readings: `buffer.STYLE_STANDARD` (default) • Store the same information as standard, plus additional information, such as the ratio component of a DCV ratio measurement: `buffer.STYLE_FULL` • Store external reading buffer data: `buffer.STYLE_WRITABLE`

Details

You cannot assign user-defined reading buffers the name `defbuffer1` or `defbuffer2`.

If you create a reading buffer that has the same name as an existing user-defined buffer, the existing buffer is overwritten by the new buffer. Any data in the existing buffer is lost.

The new buffer becomes the active reading buffer.

The default fill mode of a user-defined buffer is once. You can change it to continuous.

The buffer style cannot be changed.

Writable readings are used to bring external data into the instrument. You cannot assign them to collect data from the instrument.

You can change the buffer capacity for an existing buffer through the front panel or by using the `bufferVar.capacity` command.

Example

`capTest2 = buffer.make(200)`	Creates a 200-element reading buffer named `capTest2`.

Also see

bufferVar.capacity (on page 8-25)
bufferVar.fillmode (on page 8-29)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

buffer.save()

This function saves data from the specified reading buffer to a USB flash drive.

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

Usage

```
buffer.save(bufferVar, fileName)
buffer.save(bufferVar, fileName, timeFormat)
buffer.save(bufferVar, fileName, timeFormat, start, end)
```

bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer
fileName	A string that indicates the name of the file on the USB flash drive in which to save the reading buffer
timeFormat	Defines how date and time information from the buffer is saved in the file on the USB flash drive; the values are: • Save dates, times, and fractional seconds; `buffer.SAVE_FORMAT_TIME` • Saves relative timestamps; `buffer.SAVE_RELATIVE_TIME` • Saves seconds and fractional seconds; `buffer.SAVE_RAW_TIME` • Saves timestamps; `buffer.SAVE_TIMESTAMP_TIME`
start	Defines the starting point in the buffer to start saving data
end	Defines the ending point in the buffer to stop saving data

Details

The filename must specify the full path (including `/usb1/`). If included, the file extension must be set to `.csv` (if no file extension is specified, `.csv` is added).

For options that save more than one item of time information, each item is comma-delimited. For example, the default format is `date`, `time`, and `fractional seconds` for each reading.

Examples of valid destination file names:

```
buffer.save(MyBuffer, "/usb1/myData")
buffer.save(MyBuffer, "/usb1/myData.csv")
```

The Model DMM7510 does not check for existing files when you save. Verify that you are using a unique name to avoid overwriting any existing `.csv` files on the flash drive.

Example 1

```
buffer.save(MyBuffer, "/usb1/myData.csv")
```

Save all reading and default time information from a buffer named `MyBuffer` to a file named `myData.csv` on the USB flash drive.

Example 2

```
buffer.save(MyBuffer, "/usb1/myDataRel.csv", buffer.SAVE_RELATIVE_TIME)
```

Save all readings and relative timestamps from `MyBuffer` to a file named `myDataRel.csv` on the USB flash drive.

Example 3

```
buffer.save(defbuffer1, "/usb1/defbuf1data", buffer.SAVE_RAW_TIME)
```
Save readings and raw time stamps from `defbuffer1` to a file named `defbuf1data` on the USB flash drive.

Also see

buffer.make() (on page 8-18)
buffer.saveappend() (on page 8-21)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

buffer.saveappend()

This function appends data from the reading buffer to a file on the USB flash drive.

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

Usage

```
buffer.saveappend(bufferVar, filename)
buffer.saveappend(bufferVar, filename, timeFormat)
buffer.saveappend(bufferVar, filename, timeFormat, start, end)
```

bufferVar	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
fileName	A string that indicates the name of the file on the USB flash drive in which to save the reading buffer
timeFormat	Indicates how date and time information from the buffer is saved in the file on the USB flash drive; the values are: • Save dates, times, and fractional seconds: `buffer.SAVE_FORMAT_TIME` • Saves relative timestamps: `buffer.SAVE_RELATIVE_TIME` • Saves seconds and fractional seconds: `buffer.SAVE_RAW_TIME` • Saves timestamps: `buffer.SAVE_TIMESTAMP_TIME`
start	Defines the starting point in the buffer to start saving data
end	Defines the ending point in the buffer to stop saving data

Details

If the file you specify does not exist on the USB flash drive, this command creates the file.

For options that save more than one item of time information, each item is comma-delimited. For example, the default format is date, time, and fractional seconds for each reading.

The file extension `.csv` is appended to the filename if necessary. Any file extension other than `.csv` generates an error.

The index column entry in the `.csv` file starts at 1 for each append operation.

Examples of valid destination file names:

```
buffer.saveappend(bufferVar, "/usb1/myData")
buffer.saveappend(bufferVar, "/usb1/myData.csv")
```

Invalid destination filename examples:

```
buffer.saveappend(bufferVar, "/usb1/myData.")
```

— The period is not followed by `csv`.

```
buffer.saveappend(bufferVar, "/usb1/myData.txt")
```

— The only allowed extension is `.csv`. If `.csv` is not assigned, it is automatically added.

Example 1

```
buffer.saveappend(MyBuffer, "/usb1/myData.csv")
```

Append reading and default time information from a buffer named `MyBuffer` to a file named `myData.csv` on the USB flash drive.

Example 2

```
buffer.saveappend(MyBuffer, "/usb1/myDataRel.csv", buffer.SAVE_RELATIVE_TIME)
```

Append readings and relative timestamps from `MyBuffer` to a file named `myDataRel.csv` on the USB flash drive.

Example 3

```
buffer.saveappend(defbuffer1, "/usb1/defbuf1data", buffer.SAVE_RAW_TIME, 1, 10)
```

Append the readings and raw time stamps for points `1` to `10` from `defbuffer1` to a file named `defbuf1data` on the USB flash drive.

Also see

Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)
buffer.make() (on page 8-18)
buffer.save() (on page 8-20)

buffer.write.format()

This function sets the units and number of digits of the readings that are written into the reading buffer.

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

Usage

```
buffer.write.format(bufferVar, unit, displayDigits)
```

bufferVar	A user-supplied string that indicates the name of the buffer
unit	The units for the measurement:

• buffer.UNIT_AMP	• buffer.UNIT_OHM
• buffer.UNIT_AMP_AC	• buffer.UNIT_PERCENT
• buffer.UNIT_CELSIUS	• buffer.UNIT_RATIO
• buffer.UNIT_DECIBEL	• buffer.UNIT_RECIPROCAL
• buffer.UNIT_FAHRENHEIT	• buffer.UNIT_SECOND
• buffer.UNIT_FARAD	• buffer.UNIT_VOLT
• buffer.UNIT_HERTZ	• buffer.UNIT_VOLT_AC
• buffer.UNIT_KELVIN	• buffer.UNIT_WATT
• buffer.UNIT_NONE	• buffer.UNIT_X

displayDigits	• buffer.DIGITS_3_5
	• buffer.DIGITS_4_5
	• buffer.DIGITS_5_5
	• buffer.DIGITS_6_5
	• buffer.DIGITS_7_5

Details

Defines the units and the number of digits that are reported for the data. It only affects how the data is shown in the reading buffer.

Example

```
extBuffer = buffer.make(100, buffer.STYLE_WRITABLE)
buffer.write.format(extBuffer, buffer.UNIT_WATT, buffer.DIGITS_3_5)
buffer.write.reading(extBuffer, 1)
buffer.write.reading(extBuffer, 2)
buffer.write.reading(extBuffer, 3)
buffer.write.reading(extBuffer, 4)
buffer.write.reading(extBuffer, 5)
buffer.write.reading(extBuffer, 6)
printbuffer(1, 6, extBuffer.readings, extBuffer.units)
```

Creates a 100-point reading buffer named extBuffer. Style is writable.
Set the data format to show units of watts with 3½ digit resolution.
Write 6 pieces of data into the buffer.
Print the buffer, including the readings and units.
Read the buffer. Output:
1, Watt DC, 2, Watt DC, 3, Watt DC, 4, Watt DC, 5, Watt DC, 6, Watt DC

Also see

buffer.write.reading()

This function allows you to write readings into the reading buffer.

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

Usage

```
buffer.write.reading(bufferVar, readingValue)
buffer.write.reading(bufferVar, readingValue, seconds)
buffer.write.reading(bufferVar, readingValue, seconds, fractionalSeconds)
```

bufferVar	A user-supplied string that indicates the name of the buffer
readingValue	The value that is recorded in the buffer
seconds	An integer that represents the seconds
fractionalSeconds	An integer that represents the fractional seconds

Details

This command writes the data you specify into a reading buffer. The reading buffer must be set to the writable style, which is set when you make the buffer.

Data must be added in chronological order. If the time is not specified for a reading, it is set to one integer second after the last reading. As you write the data, the front-panel home screen updates and displays the reading you entered.

Example

```
extBuffer = buffer.make(100, buffer.STYLE_WRITABLE)
buffer.write.format(extBuffer, buffer.UNIT_WATT, buffer.DIGITS_3_5)
buffer.write.reading(extBuffer, 1)
buffer.write.reading(extBuffer, 2)
buffer.write.reading(extBuffer, 3)
buffer.write.reading(extBuffer, 4)
buffer.write.reading(extBuffer, 5)
buffer.write.reading(extBuffer, 6)
printbuffer(1, 6, extBuffer.readings, extBuffer.units)
```

Creates a 100-point reading buffer named `extBuffer`. Style is writable.
Set the data format to show a units of watts with 3-½ digit resolution.
Write 6 pieces of data into the buffer.
Print the buffer, including the readings and units.
Read the buffer.
Output:
`1, Watt DC, 2, Watt DC, 3, Watt DC, 4, Watt DC, 5, Watt DC, 6, Watt DC`

Also see

buffer.make() (on page 8-18)
buffer.write.format() (on page 8-23)
Reading buffers (on page 3-12)

bufferVar.capacity

This attribute contains the number of readings a buffer can store.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R/W)	Yes	Restore configuration Instrument reset Power cycle	Not applicable	Not applicable

Usage

```
bufferCapacity = bufferVar.capacity
bufferVar.capacity = bufferCapacity
```

bufferCapacity	The maximum number of readings the buffer can store
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer

Details

This command allows you to change or view how many readings a buffer can store.

Example

```reset() testData = buffer.make(500) capTest = buffer.make(300) bufferCapacity = capTest.capacity```	Create two user-defined reading buffers: *testData* and *capTest*.
	Create a variable called `bufferCapacity` to hold the capacity of the `capTest` buffer.
`print(bufferCapacity)`	Print `bufferCapacity`.
	Output:
	`300`
`print(testData.capacity)`	Print the capacity of `testData`.
	Output:
	`500`
`testData.capacity = 600`	Changes the capacity of `testData` to 600.
`print(testData.capacity)`	Print the capacity of `testData`.
`print(defbuffer1.capacity)`	Output:
	`600`
	Print the capacity of the default buffer `defbuffer1`.
	Output:
	`100000`

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.clear()

This function clears all readings and statistics from the specified buffer.

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

Usage

```
bufferVar.clear()
```

bufferVar	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined buffer

Example

```
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 3, 0, testData)
trigger.model.initiate()
waitcomplete()
printbuffer(1, testData.n, testData)
testData.clear()
print("Readings in buffer after clear ="
     .. testData.n)
trigger.model.initiate()
waitcomplete()
printbuffer(1, testData.n, testData)
```

Create a reading buffer named testData, make three readings and store them in testData, and then view the readings.
Print number of readings in testData.
Output:
-4.5010112303956e-10, -3.9923108222095e-12, -4.5013931471161e-10
Clear the readings in testData.
Verify that there are no readings in testData.
Output:
Readings in buffer after clear = 0
Store three new readings in testData and view those when complete.
Output:
4.923509754e-07, 3.332266330e-07, 3.974883867e-07

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.dates

This attribute contains the dates of readings that are stored in the reading buffer.

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

Usage

```
date = bufferVar.dates[N]
```

date	The date of readings stored in *bufferVar* element *N*
bufferVar	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined 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 attribute contains the dates of readings that are stored in the reading buffer. The dates are formatted as month, day, year.

This is not available if the reading buffer style is set to compact.

Example

```reset()```
 ```testData = buffer.make(50)```
 ```trigger.model.load("SimpleLoop", 3, 1, testData)```
 ```trigger.model.initiate()```
 ```waitcomplete()```
 ```print(testData.dates[1])```
 ```printbuffer(1, testData.n, testData.dates)```
	Create a reading buffer named testData, configure the instrument to make three measurements, and store the readings in the buffer.
	 Print the first reading date.
	 Example output:
	 ```03/01/2013```
	 Prints the dates for readings 1 through the last reading in the buffer.
	 Example output:
	 ```03/01/2013, 03/01/2013, 03/01/2013```

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.extravalues

This attribute contains the additional values in a buffer.

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

Usage

```
extraValue = bufferVar.extravaluesN
```

extravalue	The extra values for readings
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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 contains an additional value, such as the sense voltage from a DC voltage ratio measurement. The reading buffer style must be set to full to use this option.

If there is no additional value, this returns 0.

Example

`reset()`
`testData = buffer.make(50, buffer.STYLE_FULL)`
`dmm.measure.func = dmm.FUNC_DCV_RATIO`
`dmm.measure.read(testData)`
`print(testData.extravalues[1])`
`printbuffer(1,1,testData.extravalues)`
	Reset the instrument.
	Create a reading buffer named `testData` that can hold a maximum of 50 readings and is set to the style full.
	Make a measurement and save it to the `testData` buffer.
	Print the first extra reading value.
	Example output:
	`-7.4235309424`
	`-7.4235309424`

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.fillmode

This attribute determines if a reading buffer is filled continuously or is filled once and stops.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	User-defined buffer: buffer.FILL_ONCE (0) defbuffer1: buffer.FILL_CONTINUOUS (1) defbuffer2: buffer.FILL_CONTINUOUS (1)

Usage

```
fillMode = bufferVar.fillmode
bufferVar.fillmode = fillMode
```

fillMode	Fill the buffer, then stop: `buffer.FILL_ONCE` or `0`
	Fill the buffer continuously: `buffer.FILL_CONTINUOUS` or `1`
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer

Details

When a reading buffer is set to fill once, no data is overwritten in the buffer. When the buffer is filled, no more data is stored in that buffer and new readings are discarded.

When a reading buffer is set to fill continuously, the oldest data is overwritten by the newest data after the buffer fills.

When you change the fill mode of a buffer, any data in the buffer is cleared.

Example

| ```
reset()
testData = buffer.make(50)
print(testData.fillmode)
testData.fillmode = buffer.FILL_CONTINUOUS
print(testData.fillmode)
``` | Create a reading buffer named `testData`. Print the fill mode setting for the `testData` buffer. Output: `0` Set fill mode to continuous. Print the fill mode setting for the `testData` buffer. Output: `1` |

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.formattedreadings

This attribute contains the stored readings formatted as they appear on the front-panel display.

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

Usage

```
reading = bufferVar.formattedreadings[N]
```

`reading`	Buffer reading formatted as it appears on the front-panel display for element `N`
`bufferVar`	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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 attribute outputs an array (a Lua table) of strings that contain the stored readings. The readings are shown as they would appear on the front-panel display.

Example

```
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 3, 0, testData)
trigger.model.initiate()
waitcomplete()
print(testData.formattedreadings[1])
printbuffer(1, testData.n,
    testData.formattedreadings)
```

Create a reading buffer named `testData`, configure the instrument to make three measurements, and store the readings in the buffer.
Print the first reading formatted as it appears on the front-panel display.

Example output:
`–0.0001901 V`
Print all readings in the reading buffer as they appear on the front-panel display.
Example output:
`–0.0001901 V, +000.08537 mV, –000.13050 mV`

Also see

bufferVar.readings (on page 8-34)
buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.fractionalseconds

This attribute contains the fractional second portion of the timestamp of each reading in the reading buffer.

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

Usage

```
fractionalSec = bufferVar.fractionalseconds[N]
```

fractionalSec	The fractional second portion of the timestamp to 1 ns resolution
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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 attribute is an array (a Lua table) of the fractional portion of the timestamps, in seconds, when each reading occurred. Seconds are shown as fractions.

Example

```
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 6, 0,
    testData)
trigger.model.initiate()
waitcomplete()
print(testData.fractionalseconds[1])
printbuffer(1, 6, testData.fractionalseconds)
```

Create a reading buffer named `testData` and make six measurements.

Print the fractional portion of the timestamp for the first reading in the buffer.
Example output:
`0.647118937`
Print the fractional portion of the timestamp for the first six readings in the buffer.
Example output:
`0.647118937, 0.064543,`
`    0.48196127, 0.89938724,`
`    0.316800064, 0.734218263`

Also see

[bufferVar.seconds](#) (on page 8-36)
[buffer.delete()](#) (on page 8-16)
[buffer.make()](#) (on page 8-18)
[bufferVar.clear()](#) (on page 8-26)
[print()](#) (on page 8-228)
[printbuffer()](#) (on page 8-229)
[Reading buffers](#) (on page 3-12)
[Remote buffer operation](#) (on page 3-27)

bufferVar.logstate

This attribute indicates if information events are logged when the specified reading buffer is at 0 % or 100 % filled

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	defbuffer1: buffer.ON (1) defbuffer2: buffer.ON (1) User-created buffer: buffer.OFF (0)

Usage

```
logState = bufferVar.logstate
bufferVar.logstate = logState
```

logState	Do not log information events: `buffer.OFF` or `0`
	Log information events: `buffer.ON` or `1`
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer

Details

If this is set to on, when the reading buffer is cleared (0 % filled) or full (100 % filled), an event is logged in the event log. If this is set to off, reading buffer status is not reported in the event log.

Example

``` reset() MyBuffer = buffer.make(500) print(MyBuffer.logstate) ```	Create the user-defined buffer `MyBuffer`. Print the log state of `MyBuffer`. Output: `0`

Also see

[Using the event log](#) (on page 2-152)

bufferVar.n

This attribute contains the number of readings in the specified reading buffer.

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

Usage

```
numberOfReadings = bufferVar.n
```

numberOfReadings	The number of readings stored in the reading buffer
bufferVar	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined buffer

Details

You can call this command to return the number of readings stored in the specified reading buffer.

You can use the *bufferVar*.n attribute in other commands. For example, to print all of the readings in a buffer, use the following command:

```
printbuffer(1, defbuffer1.n, defbuffer1.readings)
```

Example

```reset() testData = buffer.make(100) trigger.model.load("SimpleLoop", 3, 0, testData) trigger.model.initiate() waitcomplete() print(testData.n) print(defbuffer1.n) print(defbuffer2.n)```	Create a reading buffer named testData, configure the instrument to make three measurements, and store the readings in the buffer. Print the number of readings in testData. Output: 3 Print the number of readings in defbuffer1. Example output: 0 Print the number of readings in defbuffer2. Example output: 0

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

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	Restore configuration Instrument reset Power cycle	Not applicable	Not applicable

Usage

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

`reading`	The value of the reading in the specified reading buffer
`bufferVar`	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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

Example 1

```
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 3, 0, testData)
trigger.model.initiate()
waitcomplete()
print(testData.readings[1])
```

Create a reading buffer named `testData`, configure the instrument to make three measurements, and store the readings in the buffer.
Print the first reading in `testData`.
Output:
`-9.6420389034124e-12`

Example 2

```
printbuffer(1, 3, testData.readings)
```

For the buffer created in Example 1, print the three readings in buffer.
Output:
`-9.6420389034124e-12, -4.5509945811872e-10, -9.1078204006445e-12`

Also see

bufferVar.n (on page 8-33)
buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.relativetimestamps

This attribute contains the timestamps, in seconds, when each reading occurred, relative to the timestamp of the first entry in the reading buffer.

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

Usage

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

timestamp	The timestamp, in seconds
bufferVar	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined 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 attribute is an array (a Lua table) of timestamps, in seconds, of when each reading occurred relative to the timestamp of the first entry in the reading buffer. These timestamps are equal to the time that has lapsed for each reading since the first reading was stored in the buffer. Therefore, the relative timestamp for the first entry number in the reading buffer equals 0.

Example

```reset()```
 ```testData = buffer.make(50)```
 ```trigger.model.load("SimpleLoop", 3, 0, testData)```
 ```trigger.model.initiate()```
 ```waitcomplete()```
 ```print(testData.relativetimestamps[1])```
 ```printbuffer(1, 3, testData.relativetimestamps)```
	Create a reading buffer named testData, configure the instrument to make three measurements, and store the readings in the buffer.
	
	 Print the relative timestamp for the first reading in the buffer.
	 Example output:
	 0
	 Print the relative timestamp for the reading 1 through 3 in the buffer.
	 Example output:
	 0, 0.383541, 0.772005

Also see

[buffer.delete()](#) (on page 8-16)
[buffer.make()](#) (on page 8-18)
[bufferVar.clear()](#) (on page 8-26)
[print()](#) (on page 8-228)
[printbuffer()](#) (on page 8-229)
[Reading buffers](#) (on page 3-12)
[Remote buffer operation](#) (on page 3-27)

bufferVar.seconds

This attribute contains the timestamp of a reading in seconds, in UTC format.

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

Usage

```
nonFracSeconds = bufferVar.seconds[N]
```

`nonFracSeconds`	The nonfractional seconds portion of the timestamp when the reading was stored
`bufferVar`	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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 attribute contains the nonfractional seconds portion of the timestamp when the reading was stored in Coordinated Universal Time (UTC) format.

The nonfractional seconds portion of the timestamp gives the lowest resolution down to 1 second. To access additional resolution of a timestamp, see `bufferVar.fractionalseconds`.

Example 1

``` reset() testData = buffer.make(50) trigger.model.load("SimpleLoop", 6, 0, testData) trigger.model.initiate() waitcomplete() print(testData.seconds[1]) ```	Create a reading buffer named `testData`, configure the instrument to make six measurements, and store the readings in the buffer. Print the seconds portion of the first reading in `testData`. Example output: `1362261492`

Example 2

``
printbuffer(1, 6, testData.seconds)
``
	For the buffer created in Example 1, print the seconds portion for readings 1 to 6 in `testData`.
	Example output:
	`1362261492, 1362261492,`
	`1362261493, 1362261493,`
	`1362261493, 1362261494`

Also see

bufferVar.fractionalseconds (on page 8-31)
bufferVar.relativetimestamps (on page 8-35)

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	Restore configuration Instrument reset Power cycle	Not applicable	Not applicable

Usage

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

statusInformation	The status value when reading *N* of the specified buffer was acquired; refer to **Details**
bufferVar	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined 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 command is not available if the buffer style is set to compact or writable.

This buffer recall attribute holds 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. Refer to the following table for values.

Buffer status bits for sense measurements

Bit (hex)	Name	Decimal	Description	*statusInformation* parameter
0x0001	STAT_QUESTIONABLE	1	Measure status questionable	buffer.STAT_QUESTIONABLE
0x0006	STAT_ORIGIN	6	A/D converter from which reading originated; for the Model DMM7510, this will always be 0 (Main) or 2 (digitizer)	buffer.STAT_ORIGIN
0x0008	STAT_TERMINAL	8	Measure terminal, front is 1, rear is 0	buffer.STAT_TERMINAL
0x0010	STAT_LIMIT2_LOW	16	Measure status limit 2 low	buffer.STAT_LIMIT2_LOW
0x0020	STAT_LIMIT2_HIGH	32	Measure status limit 2 high	buffer.STAT_LIMIT2_HIGH
0x0040	STAT_LIMIT1_LOW	64	Measure status limit 1 low	buffer.STAT_LIMIT1_LOW
0x0080	STAT_LIMIT1_HIGH	128	Measure status limit 1 high	buffer.STAT_LIMIT1_HIGH
0x0100	STAT_START_GROUP	256	First reading in a group	buffer.STAT_START_GROUP

Example

| ```
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 2, 0, testData)
trigger.model.initiate()
waitcomplete()
printbuffer(1, 2, testData.statuses)
``` | Create a reading buffer named `testData`, configure the instrument to make two measurements, and store the readings in the buffer.<br>Print the status for the readings in `testData`.<br><br>Output:<br>`64, 64`<br>Indicating that the status is `STAT_LIMIT1_LOW`. |

Also see

buffer.make() (on page 8-18)
buffer.delete() (on page 8-16)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.times

This attribute contains the time when the instrument made the reading.

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

Usage

```
readingTime = bufferVar.times[N]
```

readingTime	The time of the reading in hours, minutes, and seconds
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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

Example

``
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 3, 0, testData)
trigger.model.initiate()
waitcomplete()
print(testData.times[1])
printbuffer(1, 3, testData.times)
``
	This example creates a reading buffer named `testData` and makes three measurements.
	The `print()` command outputs the time of the first reading.
	Output:
	`23:09:43`
	The `printbuffer()` command outputs the time of readings 1 to 3 in the reading buffer.
	Output:
	`23:09:43, 23:09:43, 23:09:43`

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.timestamps

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	Restore configuration Instrument reset Power cycle	Not applicable	Not applicable

Usage

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

readingTimestamp	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 name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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 attribute contains the timestamps (date, hours, minutes, seconds, and fractional seconds) of readings stored in the reading buffer.

Example 1

```
reset()
testData = buffer.make(50)
trigger.model.load("SimpleLoop", 3, 0, testData)
trigger.model.initiate()
waitcomplete()
print(testData.timestamps[1])
```

Create a reading buffer named `testData`, configure the instrument to make three measurements, and store the readings in the buffer.
Print the first reading date.
Output:
`03/01/2013 14:46:07.714614838`

Example 2

```
for x = 1, 3 do printbuffer(x, x, testData.timestamps) end
```

For the buffer created in Example 1, print the timestamps for the readings.
Output:
```
03/01/2013 14:46:07.714614838
03/01/2013 14:46:08.100468838
03/01/2013 14:46:08.487631838
```

Also see

buffer.delete() (on page 8-16)
buffer.make() (on page 8-18)
bufferVar.clear() (on page 8-26)
print() (on page 8-228)
printbuffer() (on page 8-229)
Reading buffers (on page 3-12)
Remote buffer operation (on page 3-27)

bufferVar.units

This attribute contains the unit of measure that is stored with readings in the reading buffer.

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

Usage

```
readingUnit = bufferVar.units[N]
```

readingUnit	`Volt DC`: DC voltage measurement `Volt AC`: AC voltage measurement `Amp DC`: DC current measurement `Amp AC`: AC current measurement `Ohm`: Resistance measurement `Farad`: Capacitance measurement `Hertz`: Frequency measurement `Second`: Period measurement `Ratio`: DCV ratio measurement `%`: Math is set to percent for the measurements `mX+b`: Math is set to mx+b for the measurements `Reciprocal`: Math is set to reciprocal for the measurements `Decibel`: Units are set to decibel `Celsius`: Temperature measurement `Kelvin`: Temperature measurement `Fahrenheit`: Temperature measurement `Watt DC`: Power measurement
bufferVar	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined 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 attribute contains the unit of measure that is stored with readings in the reading buffer.

Example

``
reset()
testData = buffer.make(50)
testData.fillmode = buffer.FILL_CONTINUOUS
dmm.measure.func = dmm.FUNC_DC_CURRENT
trigger.model.load("SimpleLoop", 3, 0,
    testData)
trigger.model.initiate()
waitcomplete()
printbuffer(1, testData.n, testData.units)
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
trigger.model.initiate()
waitcomplete()
printbuffer(1, testData.n, testData.units)
``
	Create a reading buffer named `testData`, configure the instrument to make three measurements, and store the readings in the buffer.
	Set the buffer to fill continuously.
	Set the measure function to current.
	Make three readings.
	Print the units for the readings.
	Output:
	`Amp DC, Amp DC, Amp DC`
	Set the measure function to voltage.
	Make three readings.
	Output:
	`Volt DC, Volt DC, Volt DC`

Also see

[buffer.delete()](#) (on page 8-16)
[buffer.make()](#) (on page 8-18)
[bufferVar.clear()](#) (on page 8-26)
[print()](#) (on page 8-228)
[printbuffer()](#) (on page 8-229)
[Reading buffers](#) (on page 3-12)
[Remote buffer operation](#) (on page 3-27)

createconfigscript()

This function creates a setup file that captures most of the present settings of the instrument.

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

Usage

```
createconfigscript(scriptName)
```

scriptName	A string that represents the name of the script that will be created

Details

If `scriptName` is set to `autoexec`, the power up script in the instrument is replaced by the new configuration script. The power up script is a script that runs automatically when the instrument is powered on.

If `scriptName` is set to the name of an existing script, an event message is returned. You must delete the existing script before using the same script name. This function does not automatically overwrite existing scripts with the same name.

Once created, the script that contains the settings can be run and edited like any other script.

Example

`createconfigscript("myConfigurationScript")`	Captures the present settings of the instrument into a script named `myConfigurationScript`.

Also see

Saving setups (on page 2-148)
script.delete() (on page 8-234)

dataqueue.add()

This function adds an entry to the data queue.

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

Usage

```
result = dataqueue.add(value)
result = dataqueue.add(value, timeout)
```

result	The resulting value of `true` or `false` based on the success of the function
value	The data item to add; `value` can be of any type
timeout	The maximum number of seconds to wait for space in the data queue

Details

You cannot use the *timeout* value when accessing the data queue from a remote node (you can only use the *timeout* value while adding data to the local data queue).

The *timeout* value is ignored if the data queue is not full.

The `dataqueue.add()` function returns `false`:

- If the timeout expires before space is available in the data queue
- If the data queue is full and a *timeout* value is not specified

If the value is a table, a duplicate of the table and any subtables is made. The duplicate table does not contain any references to the original table or to any subtables.

Example

| ```
dataqueue.clear()
dataqueue.add(10)
dataqueue.add(11, 2)
result = dataqueue.add(12, 3)
if result == false then
   print("Failed to add 12 to the dataqueue")
end
print("The dataqueue contains:")
while dataqueue.count > 0 do
   print(dataqueue.next())
end
``` | Clear the data queue.<br>Each line adds one item to the data queue.<br>Output:<br>`The dataqueue contains:`<br>`1.00000e+01`<br>`1.10000e+01`<br>`1.20000e+01` |

Also see

dataqueue.CAPACITY (on page 8-44)
dataqueue.clear() (on page 8-45)
dataqueue.count (on page 8-46)
dataqueue.next() (on page 8-46)

dataqueue.CAPACITY

This constant is the maximum number of entries that you can store in the data queue.

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

Usage

```
count = dataqueue.CAPACITY
```

count	The variable that is assigned the value of `dataqueue.CAPACITY`

Details

This constant always returns the maximum number of entries that can be stored in the data queue.

Example

`MaxCount = dataqueue.CAPACITY`
`while dataqueue.count < MaxCount do`
`   dataqueue.add(1)`
`end`
`print("There are " .. dataqueue.count`
`   .. " items in the data queue")`
	This example fills the data queue until it is full and prints the number of items in the queue.
	Output:
	`There are 128 items in the data`
	`   queue`

Also see

[dataqueue.add()](#) (on page 8-43)
[dataqueue.clear()](#) (on page 8-45)
[dataqueue.count](#) (on page 8-46)
[dataqueue.next()](#) (on page 8-46)

dataqueue.clear()

This function clears the data queue.

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

Usage

`dataqueue.clear()`

Details

This function forces all `dataqueue.add()` commands that are in progress to time out and deletes all data from the data queue.

Example

`MaxCount = dataqueue.CAPACITY`
`while dataqueue.count < MaxCount do`
`   dataqueue.add(1)`
`end`
`print("There are " .. dataqueue.count`
`   .. " items in the data queue")`
`dataqueue.clear()`
`print("There are " .. dataqueue.count`
`   .. " items in the data queue")`
	This example fills the data queue and prints the number of items in the queue. It then clears the queue and prints the number of items again.
	Output:
	`There are 128 items in the data`
	`   queue`
	`There are 0 items in the data queue`

Also see

[dataqueue.add()](#) (on page 8-43)
[dataqueue.CAPACITY](#) (on page 8-44)
[dataqueue.count](#) (on page 8-46)
[dataqueue.next()](#) (on page 8-46)

dataqueue.count

This attribute contains the number of items in the data queue.

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

Usage

```
count = dataqueue.count
```

count	The number of items in the data queue

Details

The count is updated as entries are added with `dataqueue.add()` and read from the data queue with `dataqueue.next()`. It is also updated when the data queue is cleared with `dataqueue.clear()`.

A maximum of `dataqueue.CAPACITY` items can be stored at any one time in the data queue.

Example

```
MaxCount = dataqueue.CAPACITY
while dataqueue.count < MaxCount do
   dataqueue.add(1)
end
print("There are " .. dataqueue.count
   .. " items in the data queue")
dataqueue.clear()
print("There are " .. dataqueue.count
   .. " items in the data queue")
```

This example fills the data queue and prints the number of items in the queue. It then clears the queue and prints the number of items again.
Output:
```
There are 128 items in the data queue
There are 0 items in the data queue
```

Also see

dataqueue.next()

This function removes the next entry from the data queue.

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

Usage

```
value = dataqueue.next()
value = dataqueue.next(timeout)
```

value	The next entry in the data queue
timeout	The number of seconds to wait for data in the queue

Details

If the data queue is empty, the function waits up to the *timeout* value.

If data is not available in the data queue before the *timeout* expires, the return value is `nil`.

The entries in the data queue are removed in first-in, first-out (FIFO) order.

If the value is a table, a duplicate of the original table and any subtables is made. The duplicate table does not contain any references to the original table or to any subtables.

Example

```dataqueue.clear()```
 ```for i = 1, 10 do```
 ```   dataqueue.add(i)```
 ```end```
 ```print("There are " .. dataqueue.count```
 ```   .. " items in the data queue")```
 	
 ```while dataqueue.count > 0 do```
 ```   x = dataqueue.next()```
 ```   print(x)```
 ```end```
 ```print("There are " .. dataqueue.count```
 ```   .. " items in the data queue")```
	Clears the data queue, adds ten entries, then reads the entries from the data queue. Note that your output may differ depending on the setting of `format.asciiprecision`.
	 Output:
	 ```There are 10 items in the data queue```
	 ```1.0000000e+00```
	 ```2.0000000e+00```
	 ```3.0000000e+00```
	 ```4.0000000e+00```
	 ```5.0000000e+00```
	 ```6.0000000e+00```
	 ```7.0000000e+00```
	 ```8.0000000e+00```
	 ```9.0000000e+00```
	 ```1.0000000e+01```
	 ```There are 0 items in the data queue```

Also see

dataqueue.add() (on page 8-43)
dataqueue.CAPACITY (on page 8-44)
dataqueue.clear() (on page 8-45)
dataqueue.count (on page 8-46)
format.asciiprecision (on page 8-211)

delay()

This function delays the execution of the commands that follow it.

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

Usage

```
delay(seconds)
```

seconds	The number of seconds to delay (0 to 100,000 s)

Details

The instrument delays execution of the commands for at least the specified number of seconds and fractional seconds. However, the processing time may cause the instrument to delay 5 µs to 10 µs (typical) more than the requested delay.

Example 1

``` beeper.beep(0.5, 2400) delay(0.250) beeper.beep(0.5, 2400) ```	Emit a double-beep at 2400 Hz. The sequence is 0.5 s on, 0.25 s off, 0.5 s on.

Example 2

``` dataqueue.clear() dataqueue.add(35) timer.cleartime() delay(0.5) dt = timer.gettime() print("Delay time was " .. dt) print(dataqueue.next()) ```	Clear the data queue, add 35 to it, and then delay 0.5 seconds before reading it. Output: ``` Delay time was 0.500099 35 ```

Also see

None

digio.line[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.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	digio.MODE_DIGITAL_IN

Usage

```
lineMode = digio.line[N].mode
digio.line[N].mode = lineMode
```

lineMode	The digital line control type and line mode: Digital control, input: `digio.MODE_DIGITAL_IN` Digital control, output: `digio.MODE_DIGITAL_OUT` Digital control, open-drain: `digio.MODE_DIGITAL_OPEN_DRAIN` Trigger control, input: `digio.MODE_TRIGGER_IN` Trigger control, output: `digio.MODE_TRIGGER_OUT` Trigger control, open-drain: `digio.MODE_TRIGGER_OPEN_DRAIN` Synchronous master: `digio.MODE_SYNCHRONOUS_MASTER` Synchronous acceptor: `digio.MODE_SYNCHRONOUS_ACCEPTOR`
N	The digital I/O line (1 to 6)

Details

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 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 `lineMode` 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 `lineMode` set the line as a trigger 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 `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 `trigger.digin[N].edge` attribute.

When the line is set as a synchronous acceptor, the line detects the falling-edge input triggers and automatically latches and drives the trigger line low. Asserting an output trigger releases the latched line.

When the line is set as a synchronous master, the line detects rising-edge triggers as input. For output, the line asserts a TTL-low pulse.

Example

`digio.line[1].mode = digio.MODE_TRIGGER_OUT`	Set digital I/O line 1 to be an output trigger line.

Also see

Digital I/O lines (on page 3-47)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].edge (on page 8-257)

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

N	The digital I/O line (1 to 6)

Details

This function resets the following attributes to their default values:

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

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

Example

```
-- Set the digital I/O trigger line 3 for a falling edge
digio.line[3].mode = digio.MODE_TRIGGER_OUT
trigger.digout[3].logic = trigger.LOGIC_NEGATIVE
-- Set the digital I/O trigger line 3 to have a pulsewidth of 50 microseconds.
trigger.digout[3].pulsewidth = 50e-6
-- Use digital I/O line 5 to trigger the event on line 3.
trigger.digout[3].stimulus = trigger.EVENT_DIGIO5
-- Print configuration (before reset).
print(digio.line[3].mode, trigger.digout[3].pulsewidth, trigger.digout[3].stimulus)
-- Reset the line back to factory default values.
digio.line[3].reset()
-- Print configuration (after reset).
print(digio.line[3].mode, trigger.digout[3].pulsewidth, trigger.digout[3].stimulus)
```
```
Output before reset:
digio.MODE_TRIGGER_OUT     5e-05  trigger.EVENT_DIGIO5

Output after reset:
digio.MODE_TRIGGER_IN      1e-05  trigger.EVENT_NONE
```

Also see

digio.line[N].mode (on page 8-48)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].overrun (on page 8-258)
trigger.digout[N].pulsewidth (on page 8-261)
trigger.digout[N].stimulus (on page 8-263)

digio.line[N].state

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

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

Usage

```
digio.line[N].state = state
state = digio.line[N].state
```

N	Digital I/O trigger line (1 to 6)
state	Set the line low: `digio.STATE_LOW` or `0`
	Set the line high: `digio.STATE_HIGH` or `1`

Details

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

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

Set the state to `digio.STATE_LOW` to clear the bit; set the state to `digio.STATE_HIGH` to set the bit.

Example

`digio.line[1].mode = digio.MODE_DIGITAL_OUT` `digio.line[1].state = digio.STATE_HIGH`	Sets line 1 (bit B1) of the digital I/O port high.

Also see

digio.line[N].mode (on page 8-48)
digio.readport() (on page 8-51)
digio.writeport() (on page 8-52)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].edge (on page 8-257)

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 digital I/O port. The least significant bit (bit B1) of the binary number corresponds to digital I/O line 1; bit B6 corresponds to digital I/O line 6.

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

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

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

Example

`data = digio.readport()` `print(data)`	Assume lines 2, 4, and 6 are set high when the I/O port is read. Output: `42` This is binary 101010

Also see

digio.writeport() (on page 8-52)
Digital I/O port configuration (on page 3-45)

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	The value to write to the port (0 to 63)

Details

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

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

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

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

Example

`digio.writeport(63)`	Sets digital I/O lines 1 through 6 high (binary 111111).

Also see

digio.readport() (on page 8-51)
Digital I/O port configuration (on page 3-45)

display.changescreen()

This function changes which front-panel screen is displayed.

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

Usage

```
display.changescreen(screenName)
```

screenName	The screen to display:
	• Home screen: `display.SCREEN_HOME`
	• Home screen with large readings: `display.SCREEN_HOME_LARGE_READING`
	• Reading table screen: `display.SCREEN_READING_TABLE`
	• Graph screen (opens last selected tab): `display.SCREEN_GRAPH`
	• Histogram: `display.SCREEN_HISTOGRAM`
	• FUNCTIONS swipe screen: `display.SCREEN_FUNCTIONS_SWIPE`
	• GRAPH swipe screen: `display.SCREEN_GRAPH_SWIPE`
	• SECONDARY swipe screen: `display.SCREEN_SECONDARY_SWIPE`
	• SETTINGS swipe screen: `display.SCREEN_SETTINGS_SWIPE`
	• STATISTICS swipe screen: `display.SCREEN_STATS_SWIPE`
	• USER swipe screen: `display.SCREEN_USER_SWIPE`

Example

| ```
display.clear()
display.changescreen(display.SCREEN_USER_SWIPE)
display.settext(display.TEXT1, "Batch A122")
display.settext(display.TEXT2, "Test running")
``` | Clear the USER swipe screen and switch to display the USER swipe screen. Set the first line to read "Batch A122" and the second line to display "Test running". |
|---|---|

Also see

display.settext() (on page 8-65)

display.clear()

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

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

Usage

```
display.clear()
```

Example

```display.clear()```
```display.changescreen(display.SCREEN_USER_SWIPE)```
```display.settext(display.TEXT1, "Serial number:")```
```display.settext(display.TEXT2, localnode.serialno)```
	Clear the USER swipe screen. Set the first line to read "Serial number:" and the second line to display the serial number of the instrument.

Also see

display.settext() (on page 8-65)

display.delete()

This function allows you to remove a prompt on the front-panel display that was created with `display.prompt()`.

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

Usage

```
display.delete(promptID)
```

promptID	The identifier defined by `display.prompt()`

Details

You can use this command to remove the presently displayed prompt.

Example

```
removePrompt3 = display.prompt(display.BUTTONS_NONE, "This prompt will disappear
    in 3 seconds")
delay(3)
display.delete(removePrompt3)
```

This example displays a prompt that is automatically removed in three seconds:

Also see

display.input.number()

This function allows you to create a prompt that requests a number from the user on the front-panel display.

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

Usage

```
numberEntered = display.input.number(dialogTitle)
numberEntered = display.input.number(dialogTitle, numberFormat)
numberEntered = display.input.number(dialogTitle, numberFormat, defaultValue)
numberEntered = display.input.number(dialogTitle, numberFormat, defaultValue,
    minimumValue)
numberEntered = display.input.number(dialogTitle, numberFormat, defaultValue,
    minimumValue, maximumValue)
```

numberEntered	The number that is entered from the front-panel display; `nil` if Cancel is pressed on the keypad
dialogTitle	A string that contains the text to be displayed as the title of the dialog box on the front-panel display; can be up to 32 characters
numberFormat	The format of the displayed number: • Allow integers (negative or positive) only: `display.NFORMAT_INTEGER` (default) • Allow decimal values: `display.NFORMAT_DECIMAL` • Display numbers in exponent format: `display.NFORMAT_EXPONENT` • Display numbers with prefixes before the units symbol, such as k, m, or μ: `display.NFORMAT_PREFIX`
defaultValue	The value that is initially displayed in the displayed keypad
minimumValue	The lowest value that can be entered
maximumValue	The highest value that can be entered

Details

This command prompts the instrument operator to enter a value.

The prompt is displayed until it has been responded to.

NOTE

On the prompt, the operator can move the cursor in the entry box by touching the screen. The cursor is moved to the spot where the operator touched the screen.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.rel.enable = dmm.ON
relativeoffset = display.input.number("Relative offset value",
    display.NFORMAT_INTEGER, 0, -1000, 1000)
dmm.measure.rel.level = relativeoffset
```

This example displays a number pad on the screen that defaults to 0 and allows entries from −1000 to 1000. The number that the operator enters is assigned to the relative offset level. If the operator enters a value outside of the range, an error message is displayed.

Also see

display.input.option()

This function allows you to create an option dialog box with customizable buttons on the front-panel display.

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

Usage

```
display.BUTTON_OPTIONn = display.input.option(dialogTitle, buttonTitle1,
    buttonTitle2)
display.BUTTON_OPTIONn = display.input.option(dialogTitle, buttonTitle1,
    buttonTitle2, buttonTitleN, ... buttonTitleN)
```

n	The number of the button that is selected from the front-panel display; `nil` if Cancel is pressed on the keypad; buttons are numbered top to bottom, left to right
dialogTitle	A string that contains the text to be displayed as the title of the dialog box on the front-panel display; up to 32 characters
buttonTitle1	A string that contains the name of the first button; up to 15 characters
buttonTitle2	A string that contains the name of the second button; up to 15 characters
buttonTitleN	A string that contains the names of subsequent buttons, where *N* is a number from 3 to 10; you can define up to 10 buttons; each button can be up to 15 characters

Details

Buttons are created from top to bottom, left to right. If you have more than five buttons, they are placed into two columns.

The prompt is displayed until it has been responded to. You can only send one input prompt command at a time.

Example

```
optionID = display.input.option("Select an option", "Apple", "Orange", "Papaya",
    "Pineapple", "Blueberry", "Banana", "Grapes", "Peach", "Apricot", "Guava")
print(optionID)
```

This example displays the following dialog box:

If the user selects `Peach`, the return is `display.BUTTON_OPTION8`.

Also see

display.input.number() (on page 8-56)
display.input.prompt() (on page 8-59)
display.input.string() (on page 8-60)

display.input.prompt()

This function allows you to create a prompt that accepts a user response from the front-panel display.

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

Usage

```
buttonReturn = display.input.prompt(buttonSet, dialogTitle)
```

buttonReturn	Indicates which button was pressed: • OK: `display.BUTTON_OK` • Cancel: `display.BUTTON_CANCEL` • Yes: `display.BUTTON_YES` • No: `display.BUTTON_NO`
buttonSet	The set of buttons to display: • OK button only: `display.BUTTONS_OK` • Cancel button only: `display.BUTTONS_CANCEL` • OK and Cancel buttons: `display.BUTTONS_OKCANCEL` • Yes and No buttons: `display.BUTTONS_YESNO` • Yes, No, and Cancel buttons: `display.BUTTONS_YESNOCANCEL`
dialogTitle	A string that contains the text to be displayed as the title of the dialog box on the front-panel display; up to 63 characters

Details

This command waits for a user response to the prompt. You can use the text to ask questions that can be used to configure your test. The prompt is displayed until it has been responded to by the user. You can only send one input prompt command at a time.

Example

```
result = display.input.prompt(display.BUTTONS_YESNO, "Do you want to display the
    graph screen?")
if result == display.BUTTON_YES then
   display.changescreen(display.SCREEN_GRAPH)
end
```

This displays the prompt "Do you want to display the graph screen?" on the front-panel display:

If the operator selects **Yes**, the graph screen is displayed.

Also see

display.input.number() (on page 8-56)
display.input.option() (on page 8-58)
display.input.string() (on page 8-60)

display.input.string()

This function allows you to create a dialog box that requests text from the user through the front-panel display.

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

Usage

```
textEntered = display.input.string(dialogTitle)
textEntered = display.input.string(dialogTitle, textFormat)
```

textEntered	The text that is entered from the front-panel display; `nil` if Cancel is pressed on the keypad
dialogTitle	A string that contains the text to be displayed as the title of the dialog box on the front-panel display; up to 32 characters
textFormat	The format of the entered text: • Allow any characters: `display.SFORMAT_ANY` (default) • Allow both upper and lower case letters (no special characters): `display.SFORMAT_UPPER_LOWER` • Allow only upper case letters: `display.SFORMAT_UPPER` • Allow both upper and lower case letters, no special characters, no spaces, and limited to 32 characters: `display.SFORMAT_BUFFER_NAME`

Details

This command creates a prompt to the instrument operator to enter a string value.

The prompt is displayed until it has been responded to. You can only send one input prompt command at a time.

Example

```
value = display.input.string("Enter Test Name", display.SFORMAT_ANY)
print(value)
```

This example displays the prompt "Enter Test Name" and a keyboard that the operator can use to enter a response:

The return is the response from the operator.

Also see

display.lightstate

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Power cycle	Not applicable	display.STATE_LCD_50

Usage

```
brightness = display.lightstate
display.lightstate = brightness
```

brightness	The brightness of the display: • Full brightness: `display.STATE_LCD_100` • 75 % brightness: `display.STATE_LCD_75` • 50 % brightness: `display.STATE_LCD_50` • 25 % brightness: `display.STATE_LCD_25` • Display off: `display.STATE_LCD_OFF` • Display, key lights, and all indicators off: `display.STATE_BLACKOUT`

Details

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

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

NOTE

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

Example

`display.lightstate = display.STATE_LCD_50`	Set the display brightness to 50 %.

Also see

[Adjust the backlight brightness and dimmer](#) (on page 2-10)

display.prompt()

This function allows you to create an interactive dialog prompt that displays a custom message on the front-panel display.

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

Usage

```
promptID = display.prompt(buttonID, promptText)
```

promptID	A set of characters that identifies the prompt; up to 63 characters
buttonID	The type of prompt to display; choose one of the following options: • `display.BUTTONS_NONE` • `display.BUTTONS_OK` • `display.BUTTONS_CANCEL` • `display.BUTTONS_OKCANCEL` • `display.BUTTONS_YESNO` • `display.BUTTONS_YESNOCANCEL`
promptText	A string that contains the text that is displayed above the prompts

Details

This command displays buttons and text on the front panel. You can set up scripts that respond to the buttons when they are selected.

If you send `display.BUTTONS_NONE`, the operator needs to press the EXIT key to clear the message from the front-panel display. You can also use the `display.delete()` command to remove the prompt.

Only one prompt can be active at a time.

When the user presses a button, the button presses are returned as one of the following options:

• OK: `display.BUTTON_OK`

• Cancel: `display.BUTTON_CANCEL`

• Yes: `display.BUTTON_YES`

• No: `display.BUTTON_NO`

To capture return values, you need to use `displaywait.event()` to wait for the user button selection.

Example

```
reset()
trigger.model.load("SimpleLoop", 10, 0, defbuffer1)
display.prompt(display.BUTTONS_YESNO, "Would you like to make 10 DC voltage
   readings now?")
promptID, result = display.waitevent()
if result == display.BUTTON_YES then
   trigger.model.initiate()
end
display.prompt(display.BUTTONS_YESNO, "Would you like to switch to the Graph
   screen?")
promptID, result = display.waitevent()
if result == display.BUTTON_YES then
   display.changescreen(display.SCREEN_GRAPH)
end
```

Create a simple loop that will make 10 measurements and save them in default buffer 1.
Display the prompt shown here:

If the user presses Yes, the measurements are made.
If the user presses No, the measurements are not made and the message is removed.
Display the prompt "Would you like to switch to the Graph screen?"
If the user presses Yes, the Graph screen is displayed.
If the user presses No, the user remains on the present screen.

Also see

display.readingformat

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Not applicable	Nonvolatile memory	display.FORMAT_PREFIX

Usage

```
format = display.readingformat
display.readingformat = format
```

format	Use exponent format: `display.FORMAT_EXPONENT`
	Add a prefix to the units symbol, such as k, m, or µ: `display.FORMAT_PREFIX`

Details

This setting persists through `reset()` and power cycles.

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

Example

`display.readingformat = display.FORMAT_EXPONENT`	Change front-panel display to show readings in exponential format.

Also see

Setting the display format (on page 2-53)

display.settext()

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

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

Usage

```
display.settext(display.TEXT1, userDisplayText1)
display.settext(display.TEXT2, userDisplayText2)
```

userDisplayText1	String that contains the message for the top line of the USER swipe screen (up to 20 characters)
userDisplayText2	String that contains the message for the bottom line of the USER swipe screen (up to 32 characters)

Details

This command defines text messages for the USER swipe screen.

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

Example

```
display.clear()
display.changescreen(display.SCREEN_USER_SWIPE)
display.settext(display.TEXT1, "Batch A122")
display.settext(display.TEXT2, "Test running")
```

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

Also see

[display.clear()](#) (on page 8-54)
[display.changescreen()](#) (on page 8-53)

display.waitevent()

This function causes the instrument to wait for a user to respond to a prompt that was created with a prompt command.

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

Usage

```
objectID, subID = display.waitevent()
objectID, subID = display.waitevent(timeout)
```

objectID	A number that identifies the object, such as a prompt message, that is displayed on the front panel
subID	The returned value after a button is pressed on the front panel: • display.BUTTON_YES • display.BUTTON_NO • display.BUTTON_OK • display.BUTTON_CANCEL
timeout	The amount of time to wait before timing out; time is 0 to 300 s, where the default of 0 waits indefinitely

Details

This command waits until a user responds to a front-panel prompt that was created with the display.prompt() command.

Example

```
reset()
trigger.model.load("SimpleLoop", 10, 0, defbuffer1)
display.prompt(display.BUTTONS_YESNO, "Would you like to make 10 DC voltage
    readings now?")
promptID, result = display.waitevent()
if result == display.BUTTON_YES then
    trigger.model.initiate()
end
display.prompt(display.BUTTONS_YESNO, "Would you like to switch to the Graph
    screen?")
promptID, result = display.waitevent()
if result == display.BUTTON_YES then
    display.changescreen(display.SCREEN_GRAPH)
end
```

Create a simple loop that will make 10 measurements and save them in default buffer 1.
Display the prompt "Would you like to make 10 DC voltage readings now?"
If the user presses Yes, the measurements are made.
If the user presses No, the measurements are not made and the message is removed.
Display the prompt "Would you like to switch to the Graph screen?"
If the user presses Yes, the Graph screen is displayed.
If the user presses No, the user remains on the present screen.

Also see

dmm.digitize.analogtrigger.edge.level

This attribute defines the signal level that generates the trigger event for the edge trigger mode.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.digitize.analogtrigger.edge.level
dmm.digitize.analogtrigger.edge.level = value
```

value	The signal level that generates the trigger event

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command is only available when the analog trigger mode is set to edge.

The edge level can be set to any value in the active measurement range. See the Model DMM7510 specifications for more information on the resolution and accuracy of the analog trigger.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_EDGE
dmm.digitize.analogtrigger.edge.level = 5
dmm.digitize.analogtrigger.edge.slope = dmm.SLOPE_FALLING
```
Set the function to digitize voltage.
Set the analog trigger mode to edge.
Set the level to sense 5 V.
Set the level to be detected on a falling edge.

Also see

dmm.digitize.analogtrigger.edge.slope

This attribute defines the slope of the analog trigger edge.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.SLOPE_FALLING

Usage

```
value = dmm.digitize.leveltrigger.slope
dmm.digitize.leveltrigger.slope = value
```

value	Rising: dmm.SLOPE_RISING
	Falling: dmm.SLOPE_FALLING

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This is only available when the analog trigger mode is set to edge.

Rising causes a trigger event when the analog signal trends from below the analog signal level to above the level.

Falling causes a trigger event when the signal trends from above to below the level.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_EDGE
dmm.digitize.analogtrigger.edge.level = 5
dmm.digitize.analogtrigger.edge.slope = dmm.SLOPE_FALLING
```

Set the function to digitize voltage. Set the analog trigger mode to edge. Set the level to sense 5 V. Set the level to be detected on a falling edge.

Also see

dmm.digitize.analogtrigger.highfreqreject

This attribute enables or disables high frequency rejection on analog trigger events.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
setting = dmm.digitize.analogtrigger.highfreqreject
dmm.digitize.analogtrigger.highfreqreject = setting
```

setting	0 μs: dmm.OFF
	60 μs: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

False triggering around the set analog trigger level may occur with low frequency signals that are noisy, DC, or have low amplitude and slew rate during the peaks of input sine waves less than 250 Hz. High frequency rejection avoids the false triggers by the requiring the trigger event to be sustained for at least 60 μs. This behavior is similar to a low pass filter effect with a 4 kHz 3 dB bandwidth.

When high frequency rejection is on, 60 μs of additional trigger latency is incurred. You may also need to adjust the trigger levels to ensure that trigger condition is satisfied for at least 60 μs.

Example

`dmm.digitize.func = dmm.FUNC_DC_VOLTAGE` `dmm.digitize.analogtrigger.highfreqreject = dmm.ON`	Set the measure function to DC voltage and turn high frequency rejection for the analog trigger to on.

Also see

Analog triggering overview (on page 3-60)

dmm.digitize.analogtrigger.mode

This attribute configures the type of signal behavior that can generate a trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.MODE_OFF

Usage

```
setting = dmm.digitize.analogtrigger.mode
dmm.digitize.analogtrigger.mode = setting
```

setting	The mode setting: • Edge (signal crosses one level): dmm.MODE_EDGE • Pulse (two complementary edge events meet a specified time constraint): dmm.MODE_PULSE • Window (signal enters or exits a window defined by two levels): dmm.MODE_WINDOW • No analog triggering: dmm.MODE_OFF

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When edge is selected, the trigger occurs when the signal crosses a certain level. You also specify if the trigger occurs on the rising or falling edge of the signal.

When pulse is selected, the trigger occurs when a pulse passes through the specified level and meets the constraint that you set on its width. You also specify the polarity of the signal (above or below the trigger level).

When window is selected, the trigger occurs when the signal enters or exits the window defined by the low and high signal levels.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.range = 90
dmm.digitize.analogtrigger.mode = dmm.MODE_EDGE
dmm.digitize.analogtrigger.edge.level = 5
dmm.digitize.analogtrigger.edge.slope = dmm.SLOPE_FALLING
```

Set the function to digitize voltage.
Set the range to 90, which will select a range of 100 V.
Set the analog trigger mode to edge.
Set the level to sense 5 V.
Set the level to be detected on a falling edge.

Also see

[Analog triggering overview](#) (on page 3-60)

dmm.digitize.analogtrigger.pulse.condition

This attribute defines if the pulse must be greater than or less than the pulse width before a trigger is generated.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.CONDITION_GREATER

Usage

```
value = dmm.digitize.analogtrigger.pulse.condition
dmm.digitize.analogtrigger.pulse.condition = value
```

value	The setting: • The pulse width must be greater than the specified pulse width: dmm.CONDITION_GREATER • The pulse width must be less than the specified pulse width: dmm.CONDITION_LESS

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to pulse.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_PULSE
dmm.digitize.analogtrigger.pulse.level = 5
dmm.digitize.analogtrigger.pulse.width = 30e-6
dmm.digitize.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.digitize.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```

Set function to digitize voltage.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 μs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.digitize.analogtrigger.mode (on page 8-70)
dmm.digitize.func (on page 8-85)
dmm.digitize.analogtrigger.pulse.level (on page 8-72)
dmm.digitize.analogtrigger.pulse.polarity (on page 8-73)
dmm.digitize.analogtrigger.pulse.width (on page 8-74)

dmm.digitize.analogtrigger.pulse.level

This attribute defines the pulse level that generates a trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.digitize.analogtrigger.pulse.level
dmm.digitize.analogtrigger.pulse.level = value
```

value	The signal level

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to pulse.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_PULSE
dmm.digitize.analogtrigger.pulse.level = 5
dmm.digitize.analogtrigger.pulse.width = 30e-6
dmm.digitize.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.digitize.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```

Set function to digitize voltage.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 μs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.digitize.analogtrigger.mode (on page 8-70)
dmm.digitize.func (on page 8-85)
dmm.digitize.analogtrigger.pulse.condition (on page 8-71)
dmm.digitize.analogtrigger.pulse.polarity (on page 8-73)
dmm.digitize.analogtrigger.pulse.width (on page 8-74)

dmm.digitize.analogtrigger.pulse.polarity

This attribute defines the polarity of the pulse that generates an analog trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.POLARITY_ABOVE

Usage

```
value = dmm.digitize.analogtrigger.pulse.polarity
dmm.digitize.analogtrigger.pulse.polarity = value
```

value	The setting: • Above: dmm.POLARITY_ABOVE • Below: dmm.POLARITY_BELOW

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only used when analog trigger mode is pulse.

Determines if the trigger occurs when the pulse is above the defined signal level or below the defined signal level.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_PULSE
dmm.digitize.analogtrigger.pulse.level = 5
dmm.digitize.analogtrigger.pulse.width = 30e-6
dmm.digitize.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.digitize.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```

Set function to digitize voltage.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 μs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.digitize.analogtrigger.mode (on page 8-70)
dmm.digitize.func (on page 8-85)
dmm.digitize.analogtrigger.pulse.condition (on page 8-71)
dmm.digitize.analogtrigger.pulse.level (on page 8-72)
dmm.digitize.analogtrigger.pulse.width (on page 8-74)

dmm.digitize.analogtrigger.pulse.width

This attribute defines the threshold value for the pulse width.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	20 µs

Usage

```
value = dmm.digitize.analogtrigger.pulse.width
dmm.digitize.analogtrigger.pulse.width = value
```

value	The threshold value for the pulse width: 1 µs to 40 ms

Details

This option is only available when the analog trigger mode is set to pulse.

This option sets either the minimum or maximum pulse width that generates a trigger event. The value of pulse condition determines whether this value is interpreted as the minimum or maximum pulse width.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_PULSE
dmm.digitize.analogtrigger.pulse.level = 5
dmm.digitize.analogtrigger.pulse.width = 30e-6
dmm.digitize.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.digitize.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```

Set function to digitize voltage.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 µs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.digitize.analogtrigger.mode (on page 8-70)
dmm.digitize.func (on page 8-85)
dmm.digitize.analogtrigger.pulse.condition (on page 8-71)
dmm.digitize.analogtrigger.pulse.level (on page 8-72)
dmm.digitize.analogtrigger.pulse.polarity (on page 8-73)

dmm.digitize.analogtrigger.window.direction

This attribute defines if the analog trigger occurs when the signal enters or leaves the defined upper and lower analog signal level boundaries.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.DIRECTION_ENTER

Usage

```
value = dmm.digitize.analogtrigger.window.direction
dmm.digitize.analogtrigger.window.direction = value
```

value	The direction:
	• Enter: dmm.DIRECTION_ENTER
	• Leave: dmm.DIRECTION_LEAVE

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATio
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This is only available when the analog trigger mode is set to window.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_WINDOW
dmm.digitize.analogtrigger.window.levelhigh = 5
dmm.digitize.analogtrigger.window.levellow = 1
dmm.digitize.analogtrigger.window.direction = dmm.DIRECTION_LEAVE
```

Set function to digitize voltage.
Set the analog trigger mode to window.
Set the analog trigger high level to 5 V.
Set the analog trigger low level to 1 V.
Set the trigger to occur when the signal leaves the window.

Also see

Analog triggering overview (on page 3-60)
dmm.digitize.analogtrigger.mode (on page 8-70)
dmm.digitize.analogtrigger.window.levelhigh (on page 8-76)
dmm.digitize.analogtrigger.window.levellow (on page 8-77)
dmm.digitize.func (on page 8-85)

dmm.digitize.analogtrigger.window.levelhigh

This attribute defines the upper boundary of the analog trigger window.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	Digitize current: 5e-06 Digitize voltage: 0.05

Usage

```
value = dmm.digitize.analogtrigger.window.levelhigh
dmm.digitize.analogtrigger.window.levelhigh = value
```

value	The upper boundary of the window

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to window.

The high level must be greater than the low level.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_WINDOW
dmm.digitize.analogtrigger.window.levelhigh = 5
dmm.digitize.analogtrigger.window.levellow = 1
dmm.digitize.analogtrigger.window.direction = dmm.DIRECTION_LEAVE
```
Set function to digitize voltage.
Set the analog trigger mode to window.
Set the analog trigger high level to 5 V.
Set the analog trigger low level to 1 V.
Set the trigger to occur when the signal leaves the window.

Also see

dmm.digitize.analogtrigger.window.levellow

This attribute defines the lower boundary of the analog trigger window.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.digitize.leveltrigger.window.levellow
dmm.digitize.leveltrigger.window.levellow = value
```

value	The lower boundary of the window

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to window.
The low level must be less than the high level.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_WINDOW
dmm.digitize.analogtrigger.window.levelhigh = 5
dmm.digitize.analogtrigger.window.levellow = 1
dmm.digitize.analogtrigger.window.direction = dmm.DIRECTION_LEAVE
```

Set function to digitize voltage.
Set the analog trigger mode to window.
Set the analog trigger high level to 5 V.
Set the analog trigger low level to 1 V.
Set the trigger to occur when the signal leaves the window.

Also see

dmm.digitize.aperture

This attribute determines the aperture setting for the selected function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0 (Auto)

Usage

```
time = dmm.digitize.aperture
dmm.digitize.aperture = time
```

time	The time of the aperture in seconds or automatic: • Range: 1 µs to 1 ms; set in 1 µs increments • Automatic: 0 or dmm.APERTURE_AUTO

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The aperture determines how long the instrument makes measurements. The aperture is set to automatic or to a specific value in 1 µs intervals.

The aperture is the actual acquisition time of the instrument on the signal. It must be less than the set sample rate. The minimum aperture is 1 µs when the maximum sampling rate is 1,000,000 samples per second.

When the aperture is set to automatic, the aperture is equivalent to the sample rate interval.

If you specify an aperture and the value is less than 1 µs, it is rounded down to the nearest micro second resolution.

If you set a value that is longer than the sample rate interval, the instrument generates an error event.

Set the sample rate before changing the aperture.

The maximum aperture available is 1 divided by the sample rate. The aperture cannot be set to more than this value.

When automatic is selected, the aperture setting is set to the maximum value possible for the selected sample rate.

Example

| ```
dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT
dmm.digitize.samplerate = 1000000
dmm.digitize.aperture = 0
dmm.digitize.count = 1
print(dmm.digitize.read())
``` | Set the digitize function to measure current. Set the sample rate to 1,000,000, with a count of 1, and automatic aperture. Make a digitize measurement. |

Also see

Digitize functions (on page 2-124)
dmm.digitize.func (on page 8-85)
dmm.digitize.samplerate (on page 8-110)
dmm.measure.aperture (on page 8-125)

dmm.digitize.count

This attribute sets the number of measurements to digitize when a measurement is requested.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	10,000

Usage

```
count = dmm.digitize.count
dmm.digitize.count = count
```

count	The number of measurements to make (1 to 55,000,000)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The digitizer makes the number of readings set by this command in the time set by the sample rate. This command does not affect the trigger model.

Example

| ```
dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT
dmm.digitize.aperture = 0
dmm.digitize.samplerate = 1000000
dmm.digitize.count = 10
print(dmm.digitize.read())
``` | Set the digitize function to measure current. Set the sample rate to 1,000,000, with a count of 10, and automatic aperture. Make a digitize measurement. Example output: −0.0039799990218 |

Also see

Digitize functions (on page 2-124)
dmm.digitize.aperture (on page 8-78)
dmm.digitize.samplerate (on page 8-110)

dmm.digitize.coupling.acfilter

This attribute selects the instrument settling time when coupling is set to AC.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.AC_FILTER_SLOW

Usage

```
type = dmm.digitize.coupling.acfilter
dmm.digitize.coupling.acfilter = type
```

type	Type of AC coupling filter: • Slow (800 ms): dmm.AC_FILTER_SLOW • Fast (80 ms): dmm.AC_FILTER_FAST

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This option is only used when digitize signal coupling is set to AC.

When the signal coupling is set to AC, there may still be some DC signal content that comes in with the AC signal. To allow this signal to settle out, you can set AC coupling filter to slow. When the filter is set to slow, the instrument adds an 800 ms delay before making measurements.

When the AC coupling filter is set to fast, the instrument adds an 80 ms delay before making measurements. Set the AC filter to fast for faster settling when measuring rapidly changing inputs. For most digitize voltage measurements, the 80 ms delay is enough time for the range to settle.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.coupling.type = dmm.COUPLING_AC
dmm.digitize.coupling.acfilter = dmm.AC_FILTER_FAST
```

Set the measure function to digitize voltage.
Set the coupling type to AC.
Set the filter to fast.

Also see

DC and AC coupling (on page 2-129)
dmm.digitize.coupling.acfrequency (on page 8-81)
dmm.digitize.coupling.type (on page 8-82)

dmm.digitize.coupling.acfrequency

This attribute allows you to optimize the amplitude to compensate for signal loss across the coupling capacitor when AC coupling is selected.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1000

Usage

```
value = dmm.digitize.coupling.acfrequency
dmm.digitize.coupling.acfrequency = value
```

value	The frequency: 3 Hz to 1 MHz

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command is only used when the digitize coupling type is set to AC.

For example, if you are measuring a 50 Hz signal, you could set this to 50 Hz to compensate for voltage drop across the coupling capacitor.

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE`
`dmm.digitize.coupling.type = dmm.COUPLING_AC`
`dmm.digitize.coupling.acfrequency = 50`
	Set the digitize function to voltage.
	Set the coupling type to AC.
	Set the frequency to 500 Hz.

Also see

dmm.digitize.coupling.acfilter (on page 8-80)
dmm.digitize.coupling.type (on page 8-82)

dmm.digitize.coupling.type

This attribute determines if AC or DC signal coupling is used.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.COUPLING_DC

Usage

```
couplingType = dmm.digitize.coupling.type
dmm.digitize.coupling.type = couplingType
```

couplingType	The type of coupling: • AC: dmm.COUPLING_AC • DC: dmm.COUPLING_DC

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command selects the type of input coupling that is used for the selected function.

When DC is selected, the instrument measures AC and DC components of the signal. When AC is selected, the instrument only measures the AC components of the signal.

If AC coupling is selected, you can change input impedance settings, but they do not take effect until DC coupling is selected.

Example

dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE dmm.digitize.coupling.type = dmm.COUPLING_AC	Set the digitize voltage coupling to AC.

Also see

dmm.digitize.dbreference

This attribute defines the decibel (dB) reference setting for the DMM in volts.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
value = dmm.digitize.dbreference
dmm.digitize.dbreference = value
```

value	1e-7 V to 1000 V

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This value only applies when the unit setting for the function is set to decibels.

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE`
`dmm.digitize.unit = dmm.UNIT_DB`
`dmm.digitize.dbreference = 5`
	Sets the units to decibel and sets the dB reference to 5 for DC volts.

Also see

dmm.digitize.unit (on page 8-111)

dmm.digitize.displaydigits

This attribute describes the number of digits that are displayed on the front panel for the selected function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.DIGITS_4_5

Usage

```
value = dmm.digitize.displaydigits
dmm.digitize.displaydigits = value
```

value	• 6½ display digits: dmm.DIGITS_6_5 • 5½ display digits: dmm.DIGITS_5_5 • 4½ display digits: dmm.DIGITS_4_5 • 3½ display digits: dmm.DIGITS_3_5

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

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

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

To change the number of digits returned in a remote command reading, use `format.asciiprecision`.

Example

dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT dmm.digitize.displaydigits = dmm.DIGITS_3_5	Set the instrument to use the digitize current measure function. Set the front panel to display 3½ digits.

Also see

dmm.measure.displaydigits (on page 8-143)
format.asciiprecision (on page 8-211)

dmm.digitize.func

This attribute determines which digitize function is active.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Nonvolatile memory	dmm.FUNC_NONE

Usage

```
value = dmm.digitize.func
dmm.digitize.func = value
```

value	The digitize measurement function to make active:
	• Current: dmm.FUNC_DIGITIZE_CURRENT
	• Voltage: dmm.FUNC_DIGITIZE_VOLTAGE
	• No digitize function selected (read only): dmm.FUNC_NONE

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

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

If a basic (non-digitize) measurement function is selected, this returns dmm.FUNC_NONE. The none setting is automatically made if you select a function with dmm.measure.func or through the options from the front-panel Measure Functions tab.

Example

dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT	Set the measurement function to digitize current.

Also see

Digitize functions (on page 2-124)
dmm.measure.func (on page 8-150)

dmm.digitize.inputimpedance

This attribute determines when the 10 MΩ input divider is enabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.IMPEDANCE_10M

Usage

```
setting = dmm.digitize.inputimpedance
dmm.digitize.inputimpedance = setting
```

setting	10 MΩ for all ranges: dmm.IMPEDANCE_10M
	Automatic: dmm.IMPEDANCE_AUTO

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Automatic input impedance provides the lowest measure noise with the highest isolation on the device under test (DUT). When automatic input impedance is selected, the 100 mV to 10 V voltage ranges have more than 10 GΩ input impedance. For the 100 V and 1000 V ranges, a 10 MΩ input divider is placed across the HI and LO input terminals.

When the input impedance is set to 10 MΩ, the 100 mV to 1000 V ranges have a 10 MΩ input divider across the HI and LO input terminals. The 10 M impedance provides stable measurements when the terminals are open (approximately 100 μV at 1 PLC).

Choosing automatic input impedance is a balance between achieving low DC voltage noise on the 100 mV and 1 V ranges and optimizing measurement noise due to charge injection. The Model DMM7510 is optimized for low noise and charge injection when the DUT has less than 100 KΩ input resistance. When the DUT input impedance is more than 100 K, selecting an input impedance of 10 MΩ optimizes the measurement for lowest noise on the 100 mV and 1 V ranges. You can achieve short-term low noise and low charge injection on the 100 mV and 1 V ranges with autozero off. For the 10 V to 1000 V ranges, both input impedance settings achieve low charge injection.
The input impedance setting is only available when coupling is set to DC.

Example

dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE dmm.digitize.inputimpedance = dmm.IMPEDANCE_AUTO	Set input impedance to be set automatically when the digitize voltage function is selected.

Also see

dmm.digitize.coupling.type (on page 8-82)

dmm.digitize.limit[Y].audible

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.AUDIBLE_NONE

Usage

```
value = dmm.digitize.limit[Y].audible
dmm.digitize.limit[Y].audible = value
```

value	When the beeper sounds: • Never: dmm.AUDIBLE_NONE • On test failure: dmm.AUDIBLE_FAIL • On test pass: dmm.AUDIBLE_PASS
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The tone and length of beeper cannot be adjusted.

Example

See dmm.digitize.limit[Y].low.value (on page 8-95) for an example of how to use this command.

Also see

dmm.digitize.limit[Y].enable (on page 8-90)

dmm.digitize.limit[Y].autoclear

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.ON

Usage

```
value = dmm.digitize.limit[Y].autoclear
dmm.digitize.limit[Y].autoclear = value
```

value	The auto clear setting: • Disable: dmm.OFF • Enable: dmm.ON
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

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

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

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE` `dmm.digitize.limit[1].autoclear = dmm.OFF`	Turns off autoclear for limit 1 when measuring digitize voltage.

Also see

dmm.digitize.limit[Y].enable (on page 8-90)

dmm.digitize.limit[Y].clear()

This attribute clears the results of the limit test defined by Y.

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

Usage

```
dmm.digitize.limit[Y].clear()
```

Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

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

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE` `dmm.digitize.limit[1].clear()`	Set the digitize function to voltage. Clear the results of limit test 1.

Also see

Digitize functions (on page 2-124)
dmm.digitize.limit[Y].autoclear (on page 8-88)
dmm.digitize.func (on page 8-85)

dmm.digitize.limit[Y].enable

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

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
state = dmm.digitize.limit[Y].enable
dmm.digitize.limit[Y].enable = state
```

state	Limit Y testing: • Disable: dmm.OFF • Enable: dmm.ON
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

Example

This example enables limits 1 and 2 for digitize voltage measurements. Limit 1 is checking for readings to be between 3 and 5 V, while limit 2 is checking for the readings to be between 1 and 7 V. The auto clear feature is disabled, so if any reading is outside these limits, the corresponding fail is 1. Therefore, if any one of the fails is 1, analyze the reading buffer data to find out which reading failed the limits.

```
reset()
-- set the instrument to measure digitized voltage
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
-- set the range to 10 V
dmm.digitize.range = 10
-- disable auto clearing for limit 1
dmm.digitize.limit[1].autoclear = dmm.OFF
-- set high limit on 1 to fail if reading exceeds 5 V
dmm.digitize.limit[1].high.value = 5
-- set low limit on 1 to fail if reading is less than 3 V
dmm.digitize.limit[1].low.value = 3
-- enable limit 1 checking for digitized voltage measurements
dmm.digitize.limit[1].enable = dmm.ON
-- disable auto clearing for limit 2
dmm.digitize.limit[2].autoclear = dmm.OFF
-- set high limit on 2 to fail if reading exceeds 7 V
dmm.digitize.limit[2].high.value = 7
-- set low limit on 2 to fail if reading is less than 1 V
dmm.digitize.limit[2].low.value = 1
--- set the beeper to sound if the reading exceeds the limits for limit 2
dmm.digitize.limit[2].audible = dmm.AUDIBLE_FAIL
-- enable limit 2 checking for digitized voltage measurements
dmm.digitize.limit[2].enable = dmm.ON
-- set the measure count to 50
dmm.digitize.count = 50
-- create a reading buffer that can store 100 readings
LimitBuffer = buffer.make(100)
-- make 50 readings and store them in LimitBuffer
dmm.digitize.read(LimitBuffer)
-- Check if any of the 50 readings were outside of the limits
print("limit 1 results = " .. dmm.digitize.limit[1].fail)
print("limit 2 results = " .. dmm.digitize.limit[2].fail)
-- clear limit 1 conditions
dmm.digitize.limit[1].clear()
-- clear limit 2 conditions
dmm.digitize.limit[2].clear()
```

Example output that shows all readings are within limit values (all readings between 3 V and 5 V):
```
limit 1 results = dmm.FAIL_NONE
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 high values (a 6 V reading would cause this condition or a reading greater than 5 V but less than 7 V):
```
limit 1 results = dmm.FAIL_HIGH
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 and 2 low values (a 0.5 V reading would cause this condition or a reading less than 1 V):
```
limit 1 results = dmm.FAIL_LOW
limit 2 results = dmm.FAIL_LOW
```

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.limit[Y].low.value (on page 8-95)
dmm.digitize.limit[Y].high.value (on page 8-94)

dmm.digitize.limit[Y].fail

This attribute queries the results of a limit test.

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

Usage

```
value = dmm.digitize.limit[Y].fail
```

value	The results of the limit test for limit Y: • dmm.FAIL_NONE: Test passed; measurement under or equal to the high limit • dmm.FAIL_HIGH: Test failed; measurement exceeded high limit • dmm.FAIL_LOW: Test failed; measurement exceeded low limit • dmm.FAIL_BOTH: Test failed; measurement exceeded both limits
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

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

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

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

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

If the readings are stored in a reading buffer, you can use the *bufferVar*.statuses command to see the results.

Example

This example enables limits 1 and 2 for digitize voltage measurements. Limit 1 is checking for readings to be between 3 and 5 V, while limit 2 is checking for the readings to be between 1 and 7 V. The auto clear feature is disabled, so if any reading is outside these limits, the corresponding fail is 1. Therefore, if any one of the fails is 1, analyze the data in the reading buffer to determine which reading failed the limits.

```
reset()
-- set the instrument to digitize voltage
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
-- set the range to 10 V
dmm.digitize.range = 10
-- disable auto clearing for limit 1
dmm.digitize.limit[1].autoclear = dmm.OFF
-- set high limit on 1 to fail if reading exceeds 5 V
dmm.digitize.limit[1].high.value = 5
-- set low limit on 1 to fail if reading is less than 3 V
dmm.digitize.limit[1].low.value = 3
-- enable limit 1 checking for digitize voltage measurements
dmm.digitize.limit[1].enable = dmm.ON
-- disable auto clearing for limit 2
dmm.digitize.limit[2].autoclear = dmm.OFF
-- set high limit on 2 to fail if reading exceeds 7 V
dmm.digitize.limit[2].high.value = 7
-- set low limit on 2 to fail if reading is less than 1 V
dmm.digitize.limit[2].low.value = 1
-- enable limit 2 checking for digitize voltage measurements
dmm.digitize.limit[2].enable = dmm.ON
-- set the measure count to 50
dmm.digitize.count = 50
-- create a reading buffer that can store 100 readings
LimitBuffer = buffer.make(100)
-- make 50 readings and store them in LimitBuffer
dmm.digitize.read(LimitBuffer)
-- Check if any of the 50 readings were outside of the limits
print("limit 1 results = " .. dmm.digitize.limit[1].fail)
print("limit 2 results = " .. dmm.digitize.limit[2].fail)
-- clear limit 1 conditions
dmm.digitize.limit[1].clear()
-- clear limit 2 conditions
dmm.digitize.limit[2].clear()
```

Example output that shows all readings are within limit values (all readings between 3 V and 5 V):
```
limit 1 results = dmm.FAIL_NONE
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 high values (a 6 V reading would cause this condition or a reading greater than 5 V but less than 7 V):
```
limit 1 results = dmm.FAIL_HIGH
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 and 2 low values (a 0.5 V reading would cause this condition or a reading less than 1 V):
```
limit 1 results = dmm.FAIL_LOW
limit 2 results = dmm.FAIL_LOW
```

Also see

dmm.digitize.limit[Y].enable (on page 8-90)

dmm.digitize.limit[Y].high.value

This attribute specifies the upper limit for a limit test.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Save settings Measure configuration list	1

Usage

```
highLimit = dmm.digitize.limit[Y].high.value
dmm.digitize.limit[Y].high.value = highLimit
```

highLimit	The value of the upper limit (-1e+12 to 1e+12)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

dmm.digitize.limit[Y].enable (on page 8-90)
dmm.digitize.limit[Y].low.value (on page 8-95)

dmm.digitize.limit[Y].low.value

This attribute specifies the lower limit for limit tests.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	−1

Usage

```
lowLimit = dmm.digitize.limit[Y].low.value
dmm.digitize.limit[Y].low.value = lowLimit
```

lowLimit	The low limit value of limit *Y*; the range is −1E+12 to 1E+12
Y	Limit number 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

Example

This example enables limits 1 and 2 for digitize voltage measurements. Limit 1 is checking for readings to be between 3 and 5 V, while limit 2 is checking for the readings to be between 1 and 7 V. The auto clear feature is disabled, so if any reading is outside these limits, the corresponding fail is 1. Therefore, if any one of the fails is 1, analyze the reading buffer data to find out which reading failed the limits.

```
reset()
-- set the instrument to measure digitized voltage
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
-- set the range to 10 V
dmm.digitize.range = 10
-- disable auto clearing for limit 1
dmm.digitize.limit[1].autoclear = dmm.OFF
-- set high limit on 1 to fail if reading exceeds 5 V
dmm.digitize.limit[1].high.value = 5
-- set low limit on 1 to fail if reading is less than 3 V
dmm.digitize.limit[1].low.value = 3
-- enable limit 1 checking for digitized voltage measurements
dmm.digitize.limit[1].enable = dmm.ON
-- disable auto clearing for limit 2
dmm.digitize.limit[2].autoclear = dmm.OFF
-- set high limit on 2 to fail if reading exceeds 7 V
dmm.digitize.limit[2].high.value = 7
-- set low limit on 2 to fail if reading is less than 1 V
dmm.digitize.limit[2].low.value = 1
--- set the beeper to sound if the reading exceeds the limits for limit 2
dmm.digitize.limit[2].audible = dmm.AUDIBLE_FAIL
-- enable limit 2 checking for digitized voltage measurements
dmm.digitize.limit[2].enable = dmm.ON
-- set the measure count to 50
dmm.digitize.count = 50
-- create a reading buffer that can store 100 readings
LimitBuffer = buffer.make(100)
-- make 50 readings and store them in LimitBuffer
dmm.digitize.read(LimitBuffer)
-- Check if any of the 50 readings were outside of the limits
print("limit 1 results = " .. dmm.digitize.limit[1].fail)
print("limit 2 results = " .. dmm.digitize.limit[2].fail)
-- clear limit 1 conditions
dmm.digitize.limit[1].clear()
-- clear limit 2 conditions
dmm.digitize.limit[2].clear()
```

Example output that shows all readings are within limit values (all readings between 3 V and 5 V):
```
limit 1 results = dmm.FAIL_NONE
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 high values (a 6 V reading would cause this condition or a reading greater than 5 V but less than 7 V):
```
limit 1 results = dmm.FAIL_HIGH
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 and 2 low values (a 0.5 V reading would cause this condition or a reading less than 1 V):
```
limit 1 results = dmm.FAIL_LOW
limit 2 results = dmm.FAIL_LOW
```

Also see

dmm.digitize.limit[Y].autoclear (on page 8-88)
dmm.digitize.limit[Y].clear() (on page 8-89)
dmm.digitize.limit[Y].enable (on page 8-90)
dmm.digitize.limit[Y].fail (on page 8-92)
dmm.digitize.limit[Y].high.value (on page 8-94)

dmm.digitize.math.enable

This attribute enables or disables math operations on measurements for the selected measurement function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
value = dmm.digitize.math.enable
dmm.digitize.math.enable = value
```

value	The math enable setting: • Disable: dmm.OFF • Enable: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

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

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.math.format = dmm.MATH_PERCENT
dmm.digitize.count = 1
dmm.digitize.math.percent = dmm.digitize.read()
dmm.digitize.math.enable = dmm.ON
dmm.digitize.count = 5
MathBuffer = buffer.make(100)
dmm.digitize.read(MathBuffer)
printbuffer(1, MathBuffer.n, MathBuffer.formattedreadings)
dmm.digitize.count = 1
for x = 1, 3 do
    print(dmm.digitize.read(MathBuffer))
end
```

Configure the instrument for digitize voltage.
Set math format to percent.
Acquire 1 reading to use as the relative percent value.
Take 5 readings with percent math enabled and store them in a buffer called `MathBuffer` that can store 100 readings.

Take three additional readings.

Sample output assuming no load was connected to the instrument:
```
–100.00 %, –100.00 %, –100.00 %, –100.00 %, –100.00 %
–100.00058257
–99.999126228
–99.998932056
```

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.math.format (on page 8-98)

dmm.digitize.math.format

This attribute specifies which math operation is performed on measurements when math operations are enabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.MATH_PERCENT

Usage

```
operation = dmm.digitize.math.format
dmm.digitize.math.format = operation
```

value	Math operation to be performed on measurements: • y = mx+b: dmm.MATH_MXB • Percent: dmm.MATH_PERCENT • Reciprocal: dmm.MATH_RECIPROCAL

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This specifies which math operation is performed on measurements for the selected measurement function.

You can choose one of the following math operations:

- **y = mx+b**: Manipulate normal display readings by adjusting the m and b factors.
- **Percent**: Displays measurements as the percentage of deviation from a specified reference constant.
- **Reciprocal**: The reciprocal math operation displays measurement values as reciprocals. The displayed value is 1/X, where X is the measurement value (if relative offset is being used, this is the measured value with relative offset applied).

Math calculations are applied to the input signal after relative offset and before limit tests.

Example

dmm.digitize.func = dmm.FUNC_DC_VOLTAGE dmm.digitize.math.format = dmm.MATH_RECIPROCAL dmm.digitize.math.enable = dmm.ON	Enables the reciprocal math operation on digitize voltage measurements.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.math.enable (on page 8-97)

dmm.digitize.math.mxb.bfactor

This attribute specifies the offset, b, for the y = mx + b operation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
offsetFactor = dmm.digitize.math.mxb.bfactor
dmm.digitize.math.mxb.bfactor = offsetFactor
```

offsetFactor	The offset for the y = mx + b operation; the valid range is −1e12 to +1e12

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This attribute specifies the offset (b) for an mx + b operation.

The mx + b math operation lets you manipulate normal display readings (x) mathematically according to the following calculation:

$$y = mx + b$$

Where:

- *y* is the displayed result
- *m* is a user-defined constant for the scale factor
- *x* is the measurement reading (if you are using a relative offset, this is the measurement with relative offset applied)
- *b* is the user-defined constant for the offset factor

Example

```dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT```
```dmm.digitize.math.format = dmm.MATH_MXB```
```dmm.digitize.math.mxb.mfactor = 0.80```
```dmm.digitize.math.mxb.bfactor = 42```
```dmm.digitize.math.enable = dmm.ON```
	Set the measurement function to voltage.
	Set the scale factor for the mx +b operation to 0.80.
	Set the offset factor to 42.
	Enable the math function.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.math.enable (on page 8-97)
dmm.digitize.math.format (on page 8-98)
dmm.digitize.math.mxb.mfactor (on page 8-101)

dmm.digitize.math.mxb.mfactor

This attribute specifies the scale factor, m, for the y = mx + b math operation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
scaleFactor = dmm.digitize.math.mxb.mfactor
dmm.digitize.math.mxb.mfactor = scaleFactor
```

scaleFactor	The scale factor; the valid range is −1e12 to +1e12

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command sets the scale factor (m) for an mx + b operation for the selected measurement function.

The mx + b math operation lets you manipulate normal display readings (x) mathematically according to the following calculation:

$$y = mx + b$$

Where:

- *y* is the displayed result
- *m* is a user-defined constant for the scale factor
- *x* is the measurement reading (if you are using a relative offset, this is the measurement with relative offset applied)
- *b* is the user-defined constant for the offset factor

Example

```dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT```
```dmm.digitize.math.format = dmm.MATH_MXB```
```dmm.digitize.math.mxb.mfactor = 0.80```
```dmm.digitize.math.mxb.bfactor = 50```
```dmm.digitize.math.enable = dmm.ON```
	Set the measurement function to digitize current.
	Set the scale factor for the mx +b operation to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.math.enable (on page 8-97)
dmm.digitize.math.format (on page 8-98)
dmm.digitize.math.mxb.bfactor (on page 8-100)

dmm.digitize.math.percent

This attribute specifies the reference constant that is used when math operations are set to percent.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
value = dmm.digitize.math.percent
dmm.digitize.math.percent = value
```

value	The reference used when the math operation is set to percent; the range is -1e12 to +1e12

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The percent math function displays measurements as percent deviation from a specified reference constant. The percent calculation is:

$$Percent = \left(\frac{input - reference}{reference} \right) \times 100\%$$

Where:

- *Percent* is the result
- *Input* is the measurement (if relative offset is being used, this is the relative offset value)
- *Reference* is the user-specified constant

Example

```dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT```
```dmm.digitize.math.format = dmm.MATH_PERCENT```
```dmm.digitize.math.percent = 42```
```dmm.digitize.math.enable = dmm.ON```
	Set the measurement function to current.
	Set the math operations to percent.
	Set the percentage value to 42 for voltage measurements.
	Enable math operations.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.math.enable (on page 8-97)
dmm.digitize.math.format (on page 8-98)

dmm.digitize.range

This attribute determines the positive full-scale measure range for digitizer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script	Current: 1 A Voltage: 10 V

Usage

```
value = dmm.digitize.range
dmm.digitize.range = value
```

value	Current: 10 µA to 3 A (front terminals) or 10 A (rear terminals) Voltage: 100 mV to 1000 V

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When you assign a range value, the instrument selects a fixed range that is large enough to measure the assigned value. The instrument selects the best range for measuring the maximum expected value.

For example, for digitize current measurements, if you expect a reading of approximately 9 mA, set the range to 9 mA to select the 10 mA range.

When you read this setting, you see the positive full-scale value of the measurement range that the instrument is presently using.

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE` `dmm.digitize.range = 90`	Set the range to 90 V, which selects the 100 V range

Also see

None

dmm.digitize.read()

This function makes digitize measurements, places them in a reading buffer, and returns the last reading.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
reading = dmm.digitize.read()
reading = dmm.digitize.read(bufferName)
```

reading	The last reading of the measurement process
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; if nothing is specified, the reading is stored in `defbuffer1`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

You must set the instrument to make digitize measurements before sending this command.

This command initiates measurements using the present function setting, stores the readings in a reading buffer, and returns the last reading.

This command makes the number of digitize measurements that is set by the `dmm.digitize.count` attribute.

When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

If you define a specific reading buffer, the reading buffer must exist before you make the measurement.

Example

```voltMeasBuffer = buffer.make(10000)```
```dmm.digitize.func =```
```   dmm.FUNC_DIGITIZE_VOLTAGE```
```print(dmm.digitize.read(voltMeasBuffer))```
	Create a buffer named `voltMeasBuffer`. Set the instrument to measure voltage.
	Make a measurement that is stored in the `voltMeasBuffer` and is also printed.

Also see

buffer.make() (on page 8-18)
dmm.digitize.count (on page 8-79)
dmm.digitize.unit (on page 8-111)
Reading buffers (on page 3-12)
trigger.model.load() — SimpleLoop (on page 8-292)

dmm.digitize.readwithtime()

This function initiates digitize measurements and returns the last actual measurement and time information in UTC format without using the trigger mode.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
reading, seconds, fractional = dmm.digitize.readwithtime()
dmm.digitize.readwithtime(bufferName)
```

reading	The last reading of the measurement process
seconds	Seconds in UTC format
fractional	Fractional seconds
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; if no buffer is specified, this parameter defaults to `defbuffer1`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command initiates digitize measurements using the present function setting, stores the readings in a reading buffer, and returns the last reading.

The `dmm.digitize.count` attribute determines how many measurements are performed.

When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

If you define a specific reading buffer, the reading buffer must exist before you make the measurement.

Example

`print(dmm.digitize.readwithtime())`	Print the last digitize measurement and time information in UTC format, which will look similar to: -0.0003882925875 1415795836 0.946164546

Also see

dmm.digitize.count (on page 8-79)
trigger.model.load() — SimpleLoop (on page 8-292)

dmm.digitize.rel.acquire()

This function acquires a measurement and stores it as the relative offset value.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.digitize.rel.acquire()
```

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command triggers the instrument to make a new measurement for the selected function. This measurement is then stored as the new relative offset level.

When you send this command, the instrument does not apply any math, limit test, or filter settings to the measurement, even if they are set. It is a measurement that is made as if these settings are disabled.

If an error event occurs during the measurement or overflow, the relative offset level remains at the last valid setting.

You must change to the function for which you want to acquire a value before sending this command.

The instrument must have relative offset enabled to use the acquired relative offset value.

After executing this command, you can use the dmm.digitize.rel.level attribute to see the last relative level value that was acquired or that was set.

Example

``` dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT rel_value = dmm.digitize.rel.acquire() dmm.digitize.rel.enable = dmm.ON print(rel_value) ```	Acquires a relative offset level value for voltage measurements and turns the relative offset feature on. Output the value of the offset.

Also see

dmm.digitize.rel.enable (on page 8-107)
dmm.digitize.rel.level (on page 8-108)

dmm.digitize.rel.enable

This attribute enables or disables the application of a relative offset value to the measurement.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
state = dmm.digitize.rel.enable
dmm.digitize.rel.enable = state
```

value	The setting: • Enable: dmm.ON • Disable: dmm.OFF

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When relative measurements are enabled, all subsequent measured readings are offset by the relative offset value that was calculated when you acquired the relative offset value.

Each returned measured relative reading is the result of the following calculation:

Displayed reading = Actual measured reading - Relative offset value

Example

| ```
dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT
dmm.digitize.rel.acquire()
dmm.digitize.rel.enable = dmm.ON
``` | Enables the relative measurements for AC current and uses the acquire command to set the relative level attribute. |
|---|---|

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.digitize.rel.acquire() (on page 8-106)
dmm.digitize.rel.level (on page 8-108)

dmm.digitize.rel.level

This attribute contains the relative offset value.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.digitize.rel.level
dmm.digitize.rel.level = value
```

value	Relative offset value for measurements; see **Details**

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command specifies the relative offset value that can be applied to new measurements. When relative offset is enabled, all subsequent measured readings are offset by the value that is set for this command.

You can set this value, or have the instrument acquire a value. If the instrument acquires the value, read this setting to return the value that was measured internally.

The ranges for the relative offset values are listed in the following table.

	Minimum	Maximum
DC voltage	-1000	1000
AC voltage	-700	700
DC current (rear terminals selected)	-10	10
DC current (front terminals selected)	-3	3
AC current (rear terminals selected)	-10	10
AC current (front terminals selected)	-3	3
Resistance	-1e+09	1e+09
4-wire resistance	-1e+09	1e+09
Diode	-10	10
Capacitance	-0.001	0.001
Temperature	-3310	3310
Continuity	-1000	1000
Frequency	-1e+06	1e+06
Period	-1	1
DC voltage ratio - Method set to result	-1E+12	1E+12
DC voltage ratio - Method set to parts	-1000	1000
Digitize voltage	-1000	1000
Digitize current (rear terminals selected)	-10	10
Digitize current (front terminals selected)	-3	3

NOTE

If you have math, limits, or filter operations selected, you can set the relative offset value to include the adjustments made by these operations. To include these operations, set `dmm.digitize.rel.level` to `dmm.digitize.read()`. The adjustments from these operations are not used if you use the `dmm.digitize.rel.acquire()` function to set the relative offset level.

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT`
`dmm.digitize.rel.level = dmm.digitize.read()`
`dmm.digitize.rel.enable = dmm.ON`
	Set the digitize function to DC current.
	Set the relative offset level to be the reading with any calculations included.
	Enable the relative offset.

Also see

Relative offset (on page 3-5)
dmm.digitize.rel.acquire() (on page 8-106)
dmm.digitize.rel.enable (on page 8-107)

dmm.digitize.samplerate

This attribute defines the precise acquisition rate at which the digitizing measurements are made.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1,000,000

Usage

```
readings = dmm.digitize.samplerate
dmm.digitize.samplerate = readings
```

readings	The number of readings per second: 1,000 to 1,000,000

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The sample rate determines how fast the Model DMM7510 acquires a digitized reading.

Set the sample rate before setting the aperture. If the aperture setting is too high for the selected sample rate, it is automatically adjusted to the highest aperture that can be used with the sample rate.

Example

| ```
dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT
dmm.digitize.aperture = 0
dmm.digitize.samplerate = 1000000
dmm.digitize.count = 1
print(dmm.digitize.read())
``` | Set the digitize function to measure current. Set the sample rate to 1,000,000, with a count of 1, and automatic aperture. Make a digitize measurement. |

Also see

dmm.digitize.aperture (on page 8-78)
dmm.digitize.count (on page 8-79)

dmm.digitize.unit

This attribute sets the units of measurement that are displayed on the front panel of the instrument and stored in the reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.UNIT_VOLT

Usage

```
value = dmm.digitize.unit
dmm.digitize.unit = value
```

value	Units to display: • Volts: `dmm.UNIT_VOLT` • Decibels: `dmm.UNIT_DB`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The change in measurement units is displayed when the next measurement occurs. You can only change the units for the voltage, temperature, and digitize voltage functions. Other functions have a fixed units setting that cannot be changed.

Example

`dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE` `dmm.digitize.unit = dmm.UNIT_DB`	Set the measure function to digitize voltage. Set the units to display in decibels.

Also see

[dmm.digitize.func](#) (on page 8-85)

dmm.digitize.userdelay[N]

This attribute sets a user-defined delay that you can use in the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
delayTime = dmm.digitize.userdelay[N]
dmm.digitize.userdelay[N] = delayTime
```

delayTime	The delay (0 for no delay, or 167 ns to 10,000 s)
N	The user delay to which this time applies (1 to 5)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

To use this command in a trigger model, assign the delay to the dynamic delay block.

The delay is specific to the selected function.

Example

```
dmm.digitize.func = dmm.FUNC_DIGITIZE_CURRENT
dmm.digitize.userdelay[2] = .5
trigger.model.setblock(6, trigger.BLOCK_DELAY_DYNAMIC, trigger.USER_DELAY_M2)
```

Set user delay 2 to be 0.5 s. Sets trigger model block 6 to use the delay.

Also see

trigger.model.setblock() — trigger.BLOCK_DELAY_DYNAMIC (on page 8-311)

dmm.measure.analogtrigger.edge.level

This attribute defines the signal level that generates the trigger event for the edge trigger mode.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.measure.analogtrigger.edge.level
dmm.measure.analogtrigger.edge.level = value
```

value	The signal level that generates the trigger

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command is only available when the analog trigger mode is set to edge.

The edge level can be set to any value in the active measurement range. See the Model DMM7510 specifications for more information on the resolution and accuracy of the analog trigger.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

```
dmm.measure.func = dmm.FUNC_DC_CURRENT
dmm.measure.range = 3
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_EDGE
dmm.measure.analogtrigger.edge.level = 5
dmm.measure.analogtrigger.edge.slope = dmm.SLOPE_FALLING
```

Set measure function to DC current.
Set range to 3 A.
Disable autozero.
Set the analog trigger mode to edge.
Set the analog trigger level to 2.5 A.
Set the level to be detected on a falling edge.

Also see

dmm.measure.analogtrigger.edge.slope

This attribute defines the slope of the analog trigger edge.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.SLOPE_FALLING

Usage

```
value = dmm.measure.leveltrigger.slope
dmm.measure.leveltrigger.slope = value
```

value	Rising: dmm.SLOPE_RISING Falling: dmm.SLOPE_FALLING

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This is only available when the analog trigger mode is set to edge.

Rising causes a trigger event when the analog signal trends from below the analog signal level to above the level.

Falling causes a trigger event when the signal trends from above to below the level.

Example

```
dmm.measure.func = dmm.FUNC_DC_CURRENT
dmm.measure.range = 3
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_EDGE
dmm.measure.analogtrigger.edge.level = 5
dmm.measure.analogtrigger.edge.slope = dmm.SLOPE_FALLING
```

Set measure function to DC current.
Set range to 3 A.
Disable autozero.
Set the analog trigger mode to edge.
Set the analog trigger level to 2.5 A.
Set the level to be detected on a falling edge.

Also see

Analog triggering overview (on page 3-60)
dmm.measure.analogtrigger.edge.level (on page 8-113)
dmm.measure.analogtrigger.mode (on page 8-116)

dmm.measure.analogtrigger.highfreqreject

This attribute enables or disables high frequency rejection on analog trigger events.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
setting = dmm.measure.analogtrigger.highfreqreject
dmm.measure.analogtrigger.highfreqreject = setting
```

setting	0 µs: dmm.OFF 60 µs: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

False triggering around the set analog trigger level may occur with low frequency signals that are noisy, DC, or have low amplitude and slew rate during the peaks of input sine waves less than 250 Hz. High frequency rejection avoids the false triggers by the requiring the trigger event to be sustained for at least 60 µs. This behavior is similar to a low pass filter effect with a 4 kHz 3 dB bandwidth.

When high frequency rejection is on, 60 µs of additional trigger latency is incurred. You may also need to adjust the trigger levels to ensure that trigger condition is satisfied for at least 60 µs.

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE` `dmm.measure.analogtrigger.highfreqreject = dmm.ON`	Set the measure function to DC voltage and turn high frequency rejection for the analog trigger to on.

Also see

Analog triggering overview (on page 3-60)

dmm.measure.analogtrigger.mode

This attribute configures the type of signal behavior that can generate a trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.MODE_OFF

Usage

```
setting = dmm.measure.analogtrigger.mode
dmm.measure.analogtrigger.mode = setting
```

setting	The mode setting: • Edge (signal crosses one level): `dmm.MODE_EDGE` • Pulse (two complementary edge events meet a specified time constraint): `dmm.MODE_PULSE` • Window (signal enters or exits a window defined by two levels): `dmm.MODE_WINDOW` • No analog triggering: `dmm.MODE_OFF`

Functions

`dmm.FUNC_DC_VOLTAGE`	`dmm.FUNC_RESISTANCE`	`dmm.FUNC_ACV_FREQUENCY`
`dmm.FUNC_AC_VOLTAGE`	`dmm.FUNC_4W_RESISTANCE`	`dmm.FUNC_ACV_PERIOD`
`dmm.FUNC_DC_CURRENT`	`dmm.FUNC_DIODE`	`dmm.FUNC_DCV_RATIO`
`dmm.FUNC_AC_CURRENT`	`dmm.FUNC_CAPACITANCE`	`dmm.FUNC_DIGITIZE_CURRENT`
`dmm.FUNC_TEMPERATURE`	`dmm.FUNC_CONTINUITY`	`dmm.FUNC_DIGITIZE_VOLTAGE`

Details

When edge is selected, the trigger occurs when the signal crosses a certain level. You also specify if the trigger occurs on the rising or falling edge of the signal.

When pulse is selected, the trigger occurs when a pulse passes through the specified level and meets the constraint that you set on its width. You also specify the polarity of the signal (above or below the trigger level).

When window is selected, the trigger occurs when the signal enters or exits the window defined by the low and high signal levels.

Example

```
dmm.measure.func = dmm.FUNC_DC_CURRENT
dmm.measure.range = 3
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_EDGE
dmm.measure.analogtrigger.edge.level = 5
dmm.measure.analogtrigger.edge.slope = dmm.SLOPE_FALLING
```

Set measure function to DC current.
Set range to 3 A.
Disable autozero.
Set the analog trigger mode to edge.
Set the analog trigger level to 2.5 A.
Set the level to be detected on a falling edge.

Also see

Analog triggering example with digitize function (on page 3-62)
Analog triggering overview (on page 3-60)

dmm.measure.analogtrigger.pulse.condition

This attribute defines if the pulse must be greater than or less than the pulse width before a trigger is generated.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.CONDITION_GREATER

Usage

```
value = dmm.measure.analogtrigger.pulse.condition
dmm.measure.analogtrigger.pulse.condition = value
```

value	The setting: • The pulse width must be greater than the specified pulse width: dmm.CONDITION_GREATER • The pulse width must be less than the specified pulse width: dmm.CONDITION_LESS

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to pulse.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_PULSE
dmm.measure.analogtrigger.pulse.level = 5
dmm.measure.analogtrigger.pulse.width = 30e-6
dmm.measure.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.measure.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```

Set measure function to DC voltage.
Set range to 10 V.
Disable autozero.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 μs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.measure.analogtrigger.pulse.level (on page 8-119)
dmm.measure.analogtrigger.pulse.polarity (on page 8-120)
dmm.measure.analogtrigger.pulse.width (on page 8-121)

dmm.measure.analogtrigger.pulse.level

This attribute defines the pulse level that generates a trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.measure.analogtrigger.pulse.level
dmm.measure.analogtrigger.pulse.level = value
```

value	The signal level

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to pulse.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_PULSE
dmm.measure.analogtrigger.pulse.level = 5
dmm.measure.analogtrigger.pulse.width = 30e-6
dmm.measure.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.measure.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```
Set measure function to DC voltage.
Set range to 10 V.
Disable autozero.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 µs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.measure.analogtrigger.mode (on page 8-116)
dmm.measure.analogtrigger.pulse.condition (on page 8-117)
dmm.measure.analogtrigger.pulse.polarity (on page 8-120)
dmm.measure.analogtrigger.pulse.width (on page 8-121)

dmm.measure.analogtrigger.pulse.polarity

This attribute defines the polarity of the pulse that generates an analog trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.POLARITY_ABOVE

Usage

```
value = dmm.measure.analogtrigger.pulse.polarity
dmm.measure.analogtrigger.pulse.polarity = value
```

value	The setting: • Above: dmm.POLARITY_ABOVE • Below: dmm.POLARITY_BELOW

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only used when analog trigger mode is pulse. Determines if the trigger occurs when the pulse is above the defined signal level or below the defined signal level.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_PULSE
dmm.measure.analogtrigger.pulse.level = 5
dmm.measure.analogtrigger.pulse.width = 30e-6
dmm.measure.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.measure.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```
Set measure function to DC voltage.
Set range to 10 V.
Disable autozero.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 μs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

Analog triggering overview (on page 3-60)
dmm.measure.analogtrigger.mode (on page 8-116)
dmm.measure.analogtrigger.pulse.condition (on page 8-117)
dmm.measure.analogtrigger.pulse.polarity (on page 8-120)
dmm.measure.analogtrigger.pulse.width (on page 8-121)

dmm.measure.analogtrigger.pulse.width

This attribute defines the threshold value for the pulse width.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	20 µs

Usage

```
value = dmm.measure.analogtrigger.pulse.width
dmm.measure.analogtrigger.pulse.width = value
```

value	The threshold value for the pulse width: 1 µs to 40 ms

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This option is only available when the analog trigger mode is set to pulse.

This option sets either the minimum or maximum pulse width that generates a trigger event. The value of pulse condition determines whether this value is interpreted as the minimum or maximum pulse width.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_PULSE
dmm.measure.analogtrigger.pulse.level = 5
dmm.measure.analogtrigger.pulse.width = 30e-6
dmm.measure.analogtrigger.pulse.condition = dmm.CONDITION_LESS
dmm.measure.analogtrigger.pulse.polarity = dmm.POLARITY_BELOW
```

Set measure function to DC voltage.
Set range to 10 V.
Disable autozero.
Set the analog trigger mode to pulse.
Set the analog trigger level to 5 V.
Set the analog trigger pulse width to 30 µs.
Set the condition to be detect trigger within the pulse width.
Set the trigger to occur when the pulse is below the level.

Also see

dmm.measure.analogtrigger.window.direction

This attribute defines if the analog trigger occurs when the signal enters or leaves the defined upper and lower analog signal level boundaries.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.DIRECTION_ENTER

Usage

```
value = dmm.measure.analogtrigger.window.direction
dmm.measure.analogtrigger.window.direction = value
```

value	The direction: • Enter: dmm.DIRECTION_ENTER • Leave: dmm.DIRECTION_LEAVE

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATio
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This is only available when the analog trigger mode is set to window.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_WINDOW
dmm.measure.analogtrigger.window.levelhigh = 5
dmm.measure.analogtrigger.window.levellow = 1
dmm.measure.analogtrigger.window.direction = dmm.DIRECTION_LEAVE
```

Set measure function to DC voltage. Set range to 10 V. Disable autozero. Set the analog trigger mode to window. Set the analog trigger high level to 5 V. Set the analog trigger low level to 1 V. Set the trigger to occur when the signal leaves the window.

Also see

dmm.measure.analogtrigger.mode (on page 8-116)
dmm.measure.analogtrigger.window.levelhigh (on page 8-123)
dmm.measure.analogtrigger.window.levellow (on page 8-124)

dmm.measure.analogtrigger.window.levelhigh

This attribute defines the upper boundary of the analog trigger window.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	DC current: 5e-06 DC voltage: 0.05

Usage

```
value = dmm.measure.analogtrigger.window.levelhigh
dmm.measure.analogtrigger.window.levelhigh = value
```

value	The upper boundary of the window

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATio
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to window.

The high level must be greater than the low level.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_WINDOW
dmm.measure.analogtrigger.window.levelhigh = 5
dmm.measure.analogtrigger.window.levellow = 1
dmm.measure.analogtrigger.window.direction = dmm.DIRECTION_LEAVE
```

Set measure function to DC voltage.
Set range to 10 V.
Disable autozero.
Set the analog trigger mode to window.
Set the analog trigger high level to 5 V.
Set the analog trigger low level to 1 V.
Set the trigger to occur when the signal leaves the window.

Also see

dmm.measure.analogtrigger.window.levellow

This attribute defines the lower boundary of the analog trigger window.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.measure.analogtrigger.window.levellow
dmm.measure.analogtrigger.window.levellow = value
```

value	The lower boundary of the window

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATio
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Only available when the analog trigger mode is set to window.

The high level must be greater than the low level.

To use the analog trigger with the measure functions, a range must be set (you cannot use autorange) and autozero must be disabled.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.range = 10
dmm.measure.autozero.enable = dmm.OFF
dmm.measure.analogtrigger.mode = dmm.MODE_WINDOW
dmm.measure.analogtrigger.window.levelhigh = 5
dmm.measure.analogtrigger.window.levellow = 1
dmm.measure.analogtrigger.window.direction = dmm.DIRECTION_LEAVE
```

Set measure function to DC voltage. Set range to 10 V. Disable autozero. Set the analog trigger mode to window. Set the analog trigger high level to 5 V. Set the analog trigger low level to 1 V. Set the trigger to occur when the signal leaves the window.

Also see

Analog triggering overview (on page 3-60)
dmm.measure.analogtrigger.mode (on page 8-116)
dmm.measure.analogtrigger.window.direction (on page 8-122)
dmm.measure.analogtrigger.window.levelhigh (on page 8-123)

dmm.measure.aperture

This function determines the aperture setting for the selected function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	See **Details**

Usage

```
value = dmm.measure.aperture
dmm.measure.aperture = value
```

value	The integration rate; see **Details** for ranges

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Function	Default value	Range
Voltage (AC and DC)	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Current (AC and DC)	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Resistance (2-wire and 4-wire)	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Diode	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Temperature	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s
Frequency and Period	10 ms	10 ms to 0.273 s
Voltage ratio	60 Hz: 16.67 ms 50 Hz: 20 ms	8.333 μs to 0.25 s 10 μs to 0.24 s

The aperture sets the amount of time the ADC takes when making a measurement, which is the integration period for the selected measurement function. The integration period is specified in seconds. In general, a short integration period provides a fast reading rate, while a long integration period provides better accuracy. The selected integration period is a compromise between speed and accuracy.

During the integration period, if an external trigger is sent, the trigger is ignored.

You can also set the integration rate by setting the number of power line cycles (NPLCs). Changing the NPLC value changes the aperture time and changing the aperture time changes the NPLC value.

To calculate the aperture based on the NPLC value, use the following formula.

$$\text{Aperture} = \frac{\text{NPLC}}{f}$$

where:

- Aperture is the integration rate in seconds for each integration

- NPLC is the number of power line cycles for each integration

- f is the power line frequency

If you set the NPLCs, the aperture setting changes to reflect that value. If you set the aperture, the NPLC setting is changed.

For the AC voltage and AC current functions, if the detector bandwidth setting is set to 3 Hz or 30 Hz, the aperture value is fixed and cannot be changed.

NOTE

If line synchronization is enabled, the integration period does not start until the beginning of the next power line cycle. For example, if a reading is triggered at the positive peak of a power line cycle, the integration period does not start until that power line cycle is completed. The integration period starts when the positive-going sine wave crosses zero volts.

To see the line frequency that is auto-detected by the instrument, use the `localnode.linefreq` command.

Example

`dmm.measure.aperture = 0.0035`	Set the aperture to 3.5 ms

Also see

dmm.measure.linesync (on page 8-161)
dmm.measure.nplc (on page 8-167)
localnode.linefreq (on page 8-219)

dmm.measure.autodelay

This attribute enables or disables the automatic delay that occurs before each measurement.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.DELAY_ON

Usage

```
value = dmm.measure.autodelay
dmm.measure.autodelay = value
```

value	Enable the delay: dmm.DELAY_ON Disable the delay: dmm.DELAY_OFF

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When this is enabled, a delay is added before each measurement.

Example

```dmm.measure.func = dmm.FUNC_RESISTANCE\ndmm.measure.autodelay = dmm.DELAY_ON\ndmm.measure.count = 10\nReadingBufferOne = buffer.make(1000)\ndmm.measure.read(ReadingBufferOne)```	Set the instrument to measure 2-wire ohms. Turn automatic delay on. Create a buffer named ReadingBufferOne. Set the number of measurements to 10. Make 10 measurements and store them in the reading buffer.

Also see

[delay()](#) (on page 8-47)

dmm.measure.autorange

This attribute determines if the measurement range is set manually or automatically for the selected measure function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.ON

Usage

```
state = dmm.measure.autorange
dmm.measure.autorange = state
```

state	Set the measurement range manually: dmm.OFF Set the measurement range automatically: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command determines how the range is selected.

When this command is set to off, you must set the range. If you do not set the range, the instrument remains at the range that was selected by autorange.

When this command is set to on, the instrument automatically goes to the most sensitive range to perform the measurement.

If a range is manually selected through the front panel or a remote command, this command is automatically set to off.

Auto range selects the best range in which to measure the signal that is applied to the input terminals of the instrument. When auto range is enabled, the range increases at 120 % of range and decreases occurs when the reading is <10 % of nominal range. For example, if you are on the 1 volt range and auto range is enabled, the instrument auto ranges up to the 10 volt range when the measurement exceeds 1.2 volts. It auto ranges down to the 100 mV range when the measurement falls below 1 volt.

<div style="text-align:center">NOTE</div>

When the TERMINALS switch is set to REAR and autorange is enabled, autoranging is limited to ranges up to 3 A ranges. The 10 A range is not included in the autorange algorithm.

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE` `dmm.measure.range = 0.5`	Select the measurement function to be DC voltage. The instrument selects the 1 V range.

Also see

dmm.measure.range (on page 8-170)
Ranges

dmm.measure.autozero.enable

This attribute enables or disables automatic updates to the internal reference measurements (autozero) of the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.ON

Usage

```
state = dmm.measure.autozero.enable
dmm.measure.autozero.enable = state
```

state	Disable autozero: dmm.OFF
	Enable autozero: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

To ensure the accuracy of readings, the instrument must periodically get new measurements of its internal ground and voltage reference. The time interval between updates to these reference measurements is determined by the integration aperture that is being used for measurements. The Model DMM7510 uses separate reference and zero measurements for each aperture.

By default, the instrument automatically checks these reference measurements whenever a signal measurement is made.

The time to make the reference measurements is in addition to the normal measurement time. If timing is critical, you can disable autozero to avoid this time penalty.

When autozero is set to off, the instrument may gradually drift out of specification. To minimize the drift, you can send the once command to make a reference and zero measurement immediately before a test sequence.

For AC voltage and AC current measurements where the detector bandwidth is set to 3 Hz or 30 Hz, autozero is set on and cannot be changed.

Example

dmm.measure.func = dmm.FUNC_DC_VOLTAGE dmm.measure.autozero.enable = dmm.OFF	Set autozero off for voltage measurements.

Also see

Automatic reference measurements (on page 2-147)
dmm.measure.autozero.once() (on page 8-130)
dmm.measure.nplc (on page 8-167)

dmm.measure.autozero.once()

This function causes the instrument to refresh the reference and zero measurements once.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.autozero.once()
```

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command forces a refresh of the reference and zero measurements that are used for the present aperture setting for the selected function.

When autozero is set to off, the instrument may gradually drift out of specification. To minimize the drift, you can send the once command to make a reference and zero measurement immediately before a test sequence.

If the NPLC setting is less than 0.2 PLC, sending autozero once can result in delay of more than a second.

Example

dmm.measure.autozero.once()	Do a one-time refresh of the reference and zero measurements.

Also see

Automatic reference measurements (on page 2-147)
dmm.measure.autozero.enable (on page 8-129)

dmm.measure.bias.actual

This attribute returns the amount of current the instrument is sourcing when it makes measurements.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
value = dmm.measure.bias.actual
```

value	The actual bias level

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Reads the actual amount of current that is sourced by the instrument when a measurement is made.

Example

print(dmm.measure.bias.actual)	Read the actual bias level.

Also see

dmm.measure.bias.level (on page 8-132)

dmm.measure.bias.level

This attribute selects the amount of current the instrument sources when it makes measurements.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1 mA

Usage

```
value = dmm.measure.bias.level
dmm.measure.bias.level = value
```

value	10 µA, 100 µA, 1 mA

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Selects the amount of current that is sourced by the instrument to make measurements.

Example

`dmm.measure.bias.level = 0.0001`	Set a bias level of 100 µA.

Also see

[dmm.measure.bias.actual](#) (on page 8-131)

dmm.measure.configlist.catalog()

This function returns the name of one measure configuration list that is stored on the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.configlist.catalog()
```

Details

You can use this command to retrieve the names of measure configuration lists that are stored in the instrument.

This command returns one name each time you send it. This command returns `nil` to indicate that there are no more names to return. If the command returns `nil` the first time you send it, no measure configuration lists have been created for the instrument.

Example

`print(dmm.measure.configlist.catalog())`	Request the name of one measure configuration list that is stored in the instrument. Send the command again until it returns `nil` to get all stored lists.
`print(dmm.measure.configlist.catalog())`	If there are two configuration lists on the instrument. Example output: `testMeasList`
`print(dmm.measure.configlist.catalog())`	`myMeasList`
`print(dmm.measure.configlist.catalog())`	`nil`

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.create() (on page 8-134)

dmm.measure.configlist.create()

This function creates an empty measure configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.configlist.create(listName)
```

listName	A string that represents the name of a measure configuration list

Details

This command creates an empty configuration list. To add configuration points to this list, you need to use the store command.

Configuration lists are not saved when the instrument is turned off. To save a configuration list, create a configuration script to save instrument settings, including any defined configuration lists.

Example

```
dmm.measure.configlist.create("MyMeasList")
```

Create a measure configuration list named `MyMeasList`.

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.catalog() (on page 8-133)
dmm.measure.configlist.delete() (on page 8-135)
dmm.measure.configlist.query() (on page 8-135)
dmm.measure.configlist.recall() (on page 8-137)
dmm.measure.configlist.size() (on page 8-138)
dmm.measure.configlist.store() (on page 8-139)

dmm.measure.configlist.delete()

This function deletes a measure configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.configlist.delete(listName)
dmm.measure.configlist.delete(listName, point)
```

listName	A string that represents the name of a measure configuration list
point	A number that defines a specific configuration point in the configuration list

Details

Deletes a configuration list. If the point is not specified, the entire configuration list is deleted. If the point is specified, only the specified configuration point in the list is deleted.

Example

dmm.measure.configlist.delete("myMeasList")	Delete a measure configuration list named myMeasList.
dmm.measure.configlist.delete("myMeasList", 2)	Delete configuration point 2 from the measure configuration list named myMeasList.

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.create() (on page 8-134)

dmm.measure.configlist.query()

This function returns a list of TSP commands and parameter settings that are stored in the specified configuration point.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.configlist.query(listName, point)
dmm.measure.configlist.query(listName, point, fieldSeparator)
```

listName	A string that represents the name of a measure configuration list
point	A number that defines a specific configuration point in the configuration list
fieldSeparator	String that represents the separator for the data; use one of the following: • Comma (default): , • Semicolon: ; • New line: \n

Details

This command returns data for one configuration point.

Example

```
print(dmm.measure.configlist.query("testMeasList", 2, "\n"))
```

Returns the TSP commands and parameter settings that represent the settings in configuration point 2.
Partial example output:

```
dmm.measure.func = dmm.FUNC_AC_VOLTAGE
dmm.measure.unit = dmm.UNIT_VOLT
dmm.measure.range = 10
dmm.measure.autorange = dmm.ON
dmm.measure.autozero.enable = dmm.ON
dmm.measure.autodelay = dmm.DELAY_ON
dmm.measure.bias.level is not used
dmm.measure.detectorbandwidth = dmm.DETECTBW_30HZ
dmm.measure.displaydigits = dmm.DIGITS_6_5
dmm.measure.dbreference = 1
dmm.measure.drycircuit = dmm.OFF
dmm.measure.filter.enable = dmm.OFF
dmm.measure.filter.count = 10
dmm.measure.filter.type = dmm.FILTER_REPEAT_AVG
dmm.measure.filter.window = 0.1
```

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.create() (on page 8-134)

dmm.measure.configlist.recall()

This function recalls a configuration point in a measure configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.configlist.recall(listName)
dmm.measure.configlist.recall(listName, point)
```

listName	A string that represents the name of a measure configuration list
point	A number that defines a specific configuration point in the configuration list

Details

Use this command to recall the settings stored in a specific configuration point in a specific configuration list. If you do not specify a point when you send the command, it recalls the settings stored in the first configuration point in the specified configuration list.

Each point contains the settings for the selected function. Settings for other functions are not affected when the configuration list point is recalled.

This command returns data for one configuration point.

Example

`dmm.measure.configlist.recall("MyMeasList")`	Since a point was not specified, this command recalls configuration point 1 from a configuration list named `MyMeasList`.
`dmm.measure.configlist.recall("MyMeasList", 5)`	Recalls configuration point 5 in a configuration list named `MyMeasList`.

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.create() (on page 8-134)
dmm.measure.configlist.store() (on page 8-139)

dmm.measure.configlist.size()

This function returns the size (number of configuration points) of a measure configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

dmm.measure.configlist.size(*listName*)

listName	A string that represents the name of a measure configuration list

Details

This command returns the size (number of configuration points) of a measure configuration list.

The size of the list is equal to the number of configuration points in a configuration list.

Example

print(dmm.measure.configlist.size("testMeasList"))	Returns the number of configuration points in a measure configuration list named testMeasList. Example output: 1

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.create() (on page 8-134)

dmm.measure.configlist.store()

This function stores the active measure settings into the named configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.configlist.store(listName)
dmm.measure.configlist.store(listName, index)
```

listName	A string that represents the name of a measure configuration list
index	A number that defines a specific configuration point in the configuration list

Details

Use this command to store the active measure settings to a configuration point in a configuration list. If the *index* parameter is not provided, the configuration point will append to the end of the list.

Configuration lists are not saved when the instrument is turned off. To save a configuration list, create a configuration script to save instrument settings, including any defined configuration lists.

You cannot store both digitize and measure function settings in a configuration list that is used with a trigger model.

Example

`dmm.measure.configlist.store("MyConfigList")`	Stores the active settings of the instrument to the end of the configuration list `MyConfigList`.
`dmm.measure.configlist.store("MyConfigList", 5)`	Stores the active settings of the instrument to configuration point 5 in the measure configuration list `MyConfigList`.

Also see

Configuration lists (on page 3-32)
dmm.measure.configlist.create() (on page 8-134)

dmm.measure.count

This attribute sets the number of measurements to make when a measurement is requested.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	1

Usage

```
value = dmm.measure.count
dmm.measure.count = value
```

value	The number of measurements to make when a measurement is requested (maximum 1,000,000)

Details

This command sets the number of measurements that are made when a measurement is requested. This command does not affect the trigger model.

This command sets the count for all measure functions.

If you set the count to a value that is larger than the capacity of the reading buffer and the buffer fill mode is set to continuous, the buffer wraps until the number of readings specified have occurred. The earliest readings in the count are overwritten. If the buffer is set to fill once, readings stop when the buffer is filled, even if the count is not complete.

NOTE

To get better performance from the instrument, use the Simple Loop trigger model template instead of using the count command.

Example

`dmm.measure.count = 10` `dmm.measure.read()`	Set the instrument to make 10 measurements. Request 10 measurements.

Also see

dmm.measure.read() (on page 8-172)
dmm.measure.readwithtime() (on page 8-173)
trigger.model.load() — SimpleLoop (on page 8-292)

dmm.measure.dbreference

This attribute defines the decibel (dB) reference setting for the DMM in volts.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
value = dmm.measure.dbreference
dmm.measure.dbreference = value
```

value	The decibel reference range: • DC voltage: 1e-7 V to 1000 V • AC voltage: 1e-7 V to 700 V

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This value only applies when the unit setting for the function is set to decibels.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.unit = dmm.UNIT_DB
dmm.measure.dbreference = 5
```

Sets the units to decibel and sets the dB reference to 5 for DC volts.

Also see

dmm.measure.detectorbandwidth

This attribute selects the detector bandwidth for AC current and AC voltage measurements.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	30 Hz

Usage

```
value = dmm.measure.detectorbandwidth
dmm.measure.detectorbandwidth = value
```

value	The bandwidth that is closest to the line frequency:
	• 3 Hz: dmm.DETECTBW_3HZ
	• 30 Hz: dmm.DETECTBW_30HZ
	• 300 Hz: dmm.DETECTBW_300HZ

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

You can set the detector bandwidth to improve measurement accuracy. Select the bandwidth that contains the lowest frequency component of the input signal. For example, if the lowest frequency component of your input signal is 40 Hz, use a bandwidth setting of 30 Hz.

If the bandwidth is set to 3 Hz or 30 Hz, the autozero feature is always enabled and the integration rate is fixed.

Example

`dmm.measure.func = dmm.FUNC_AC_CURRENT`
`dmm.measure.detectorbandwidth =`
`    dmm.DETECTBW_3HZ`
	Set the measure function to AC current.
	Set the bandwidth to 3 Hz.

Also see

dmm.measure.autozero.enable (on page 8-129)

dmm.measure.displaydigits

This attribute determines the number of digits that are displayed for measurements on the front panel.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	See **Functions and defaults**

Usage

```
value = dmm.measure.displaydigits
dmm.measure.displaydigits = value
```

value	3.5 digit resolution: `dmm.DIGITS_3_5` 4.5 digit resolution: `dmm.DIGITS_4_5` 5.5 digit resolution: `dmm.DIGITS_5_5` 6.5 digit resolution: `dmm.DIGITS_6_5` 7.5 digit resolution: `dmm.DIGITS_7_5`

Functions and defaults

Function	Def	Function	Def	Function	Def
`dmm.FUNC_DC_VOLTAGE`	7	`dmm.FUNC_RESISTANCE`	7	`dmm.FUNC_ACV_FREQUENCY`	6
`dmm.FUNC_AC_VOLTAGE`	6	`dmm.FUNC_4W_RESISTANCE`	7	`dmm.FUNC_ACV_PERIOD`	6
`dmm.FUNC_DC_CURRENT`	7	`dmm.FUNC_DIODE`	7	`dmm.FUNC_DCV_RATIO`	7
`dmm.FUNC_AC_CURRENT`	6	`dmm.FUNC_CAPACITANCE`	4	`dmm.FUNC_DIGITIZE_CURRENT`	4
`dmm.FUNC_TEMPERATURE`	5	`dmm.FUNC_CONTINUITY`	4	`dmm.FUNC_DIGITIZE_VOLTAGE`	4

Details

This command affects how the reading for a measurement is displayed on the front panel of the instrument. It does not affect the number of digits returned in a remote command reading. It also does not affect the accuracy or speed of measurements.

The display digits setting is saved with the function setting, so if you use another function, then return to the function for which you set display digits, the display digits setting you set previously is retained.

The change in digits occurs the next time a measurement is made.

To change the number of digits returned in a remote command reading, use `format.asciiprecision`.

> ## NOTE
> The digits for the temperature, capacitance, continuity, frequency, and period functions are always set to the default values and cannot be changed.

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE` `dmm.measure.displaydigits = dmm.DIGITS_5_5`	Set the measurement function to voltage with a front-panel display resolution of 5½.

Also see

dmm.digitize.displaydigits (on page 8-84)
format.asciiprecision (on page 8-211)

dmm.measure.drycircuit

This attribute enables or disables the dry circuit feature of the 4-wire resistance measure function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
state = dmm.measure.drycircuit
dmm.measure.drycircuit = state
```

state	To disable dry circuit, select dmm.OFF; available for all ranges
	To enable dry circuit, select dmm.ON; available for the 1 Ω to 10 kΩ ranges

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Enabling dry circuit limits the open-circuit voltage to below 20 mV, which is often required with low-glitch measurements, such as measuring switch and relay contact resistance.
When dry circuit is enabled, offset compensation is automatically enabled.

Example

dmm.measure.func = dmm.FUNC_4W_RESISTANCE dmm.measure.drycircuit = dmm.ON	Set the measure function to 4-wire resistance and enable dry circuit.

Also see

dmm.measure.offsetcompensation.enable (on page 8-168)
dmm.measure.range (on page 8-170)

dmm.measure.filter.count

This attribute sets the number of measurements that are averaged when filtering is enabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	10

Usage

```
filterCount = dmm.measure.filter.count
dmm.measure.filter.count = filterCount
```

filterCount	The number of readings required for each filtered measurement (1 to 100)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The filter count is the number of readings that are acquired and stored in the filter stack for the averaging calculation. When the filter count is larger, more filtering is done and the data is less noisy.

Example

`dmm.measure.func = dmm.FUNC_DC_CURRENT`
`dmm.measure.filter.count = 10`
`dmm.measure.filter.type = dmm.FILTER_MOVING_AVG`
`dmm.measure.filter.enable = dmm.ON`
	Set the measurement function to current.
	Set the averaging filter type to moving average, with a filter count of 10.
	Enable the averaging filter.

Also see

Filtering measurement data (on page 3-10)
dmm.measure.filter.enable (on page 8-145)
dmm.measure.filter.type (on page 8-146)

dmm.measure.filter.enable

This attribute enables or disables the averaging filter for measurements of the selected function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
filterState = dmm.measure.filter.enable
dmm.measure.filter.enable = filterState
```

`filterState`	The filter status: • Disable the filter: `dmm.OFF` • Enable the filter: `dmm.ON`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command enables or disables the averaging filter. When this is enabled, the reading returned by the instrument is an averaged value, taken from multiple measurements. The settings of the filter count and filter type for the selected measure function determines how the reading is averaged.

Example

| ```
dmm.measure.func = dmm.FUNC_DC_CURRENT
dmm.measure.filter.count = 10
dmm.measure.filter.type = dmm.FILTER_MOVING_AVG
dmm.measure.filter.enable = dmm.ON
``` | Set the measurement function to current.<br>Set the averaging filter type to moving average, with a filter count of 10.<br>Enable the averaging filter. |

Also see

Filtering measurement data (on page 3-10)
dmm.measure.filter.count (on page 8-144)
dmm.measure.filter.type (on page 8-146)

dmm.measure.filter.type

This attribute defines the type of averaging filter that is used for the selected measure function when the measurement filter is enabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.FILTER_REPEAT_AVG

Usage

```
type = dmm.measure.filter.type
dmm.measure.filter.type = type
```

type	The filter type setting: • Repeating filter: `dmm.FILTER_REPEAT_AVG` • Moving filter: `dmm.FILTER_MOVING_AVG`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When the repeating average filter is selected, a set of measurements are made. These measurements are stored in a measurement stack and averaged together to produce the averaged sample. Once the averaged sample is produced, the stack is flushed and the next set of data is used to produce the next averaged sample. This type of filter is the slowest, since the stack must be completely filled before an averaged sample can be produced.

When the moving average filter is selected, the measurements are added to the stack continuously on a first-in, first-out basis. As each measurement is made, the oldest measurement is removed from the stack. A new averaged sample is produced using the new measurement and the data that is now in the stack.

NOTE

When the moving average filter is first selected, the stack is empty. When the first measurement is made, it is copied into all the stack locations to fill the stack. A true average is not produced until the stack is filled with new measurements. The size of the stack is determined by the filter count setting.

The repeating average filter produces slower results, but produces more stable results than the moving average filter. For either method, the greater the number of measurements that are averaged, the slower the averaged sample rate, but the lower the noise error. Trade-offs between speed and noise are normally required to tailor the instrumentation to your measurement application.

Example

`dmm.measure.func = dmm.FUNC_DC_CURRENT`
`dmm.measure.filter.type = dmm.FILTER_MOVING_AVG`
`dmm.measure.filter.enable = dmm.ON`
	Set the measurement function to DC current. Set the filter type to moving average and enable filtered measurements.

Also see

dmm.measure.filter.enable (on page 8-145)

dmm.measure.filter.window

This attribute sets the window for the averaging filter that is used for measurements for the selected function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0.1

Usage

```
value = dmm.measure.filter.window
dmm.measure.filter.window = value
```

value	The filter window setting; the range is between 0 and 10 to indicate percent of range

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command selects the window size for the averaging filter.

The noise window allows a faster response time to large signal step changes. A reading that falls outside the plus or minus noise window fills the filter stack immediately.

If the noise does not exceed the selected percentage of range, the reading is based on an average of reading conversions — the normal averaging filter. If the noise does exceed the selected percentage, the reading is a single reading conversion, and new averaging starts from this point.

Example

```
dmm.measure.func = dmm.FUNC_RESISTANCE
dmm.measure.filter.type = dmm.FILTER_MOVING_AVG
dmm.measure.filter.window = 0.25
dmm.measure.filter.enable = dmm.ON
```

Set the measure function to 2-wire ohms.
Set the filter type to moving average.
Set the filter window to 0.25 and enable filtered measurements.

Also see

dmm.measure.fourrtd

This attribute defines the type of 4-wire RTD that is being used

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.RTD_PT100

Usage

```
RTDType = dmm.measure.fourrtd
dmm.measure.fourrtd = RTDType
```

RTDType	The type of 4-wire RTD:
	• PT100: `dmm.RTD_PT100`
	• PT385: `dmm.RTD_PT385`
	• PT3916: `dmm.RTD_PT3916`
	• D100: `dmm.RTD_D100`
	• F100: `dmm.RTD_F100`
	• User-specified type: `dmm.RTD_USER`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The transducer type must be set to temperature and the transducer must be set to 4-wire RTD before you can set the RTD type.

Example

`dmm.measure.func = dmm.FUNC_TEMPERATURE`
`dmm.measure.transducer = dmm.TRANS_FOURRTD`
`dmm.measure.fourrtd = dmm.RTD_PT3916`
	Set the measure function to temperature.
	Set the transducer type to 4-wire RTD.
	Set the RTD type to PT3916.

Also see

dmm.measure.rtdalpha (on page 8-179)
dmm.measure.rtdbeta (on page 8-180)
dmm.measure.rtddelta (on page 8-181)
dmm.measure.rtdzero (on page 8-182)
dmm.measure.transducer (on page 8-192)
Temperature measurements (on page 2-117)

dmm.measure.func

This attribute selects the active measure function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.FUNC_DC_VOLTAGE

Usage

```
mFunction = dmm.measure.func
dmm.measure.func = mFunction
```

mFunction	The type of measurement; see **Functions** for options

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Set this command to the type of measurement you want to make.

Reading this command returns the measure function that is presently active.

When you select a function, settings for other commands that are related to the function become active. For example, assume that:

- You selected the current function and set the math function to reciprocal.
- You changed to the voltage function and set the math function to percent.

If you return to the current function, the math function returns to reciprocal. If you then switch from the current function to the voltage function, the math function returns to percent. All attributes that begin with dmm.measure. are saved with the active measure function unless otherwise indicated in the command description.

If a digitize measurement function is active, calling this command returns dmm.FUNC_NONE. The no function setting is automatically made if you select a function with dmm.digitize.func or through the options from the front-panel Digitize Functions tab.

Example

| ```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.math.format = dmm.MATH_PERCENT
dmm.measure.math.enable = dmm.ON
dmm.measure.func = dmm.FUNC_RESISTANCE
dmm.measure.math.format = dmm.MATH_RECIPROCAL
dmm.measure.math.enable = dmm.ON
print(dmm.measure.math.format)
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
print(dmm.measure.math.format)
``` | Sets the instrument to measure voltage and set the math format to percent and enable the math functions. Set the instrument to measure resistance and set the math format to reciprocal and enable the math functions. Print the math format while the resistance measurement function is selected. The output is: `dmm.MATH_RECIPROCAL` Change the function to voltage. Print the math format. The output is: `dmm.MATH_PERCENT` |

Also see

[dmm.digitize.func](on page 8-85)

dmm.measure.inputimpedance

This attribute determines when the 10 MΩ input divider is enabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.IMPEDANCE_10M

Usage

```
setting = dmm.measure.inputimpedance
dmm.measure.inputimpedance = setting
```

setting	10 MΩ for all ranges: dmm.IMPEDANCE_10M
	Automatic: dmm.IMPEDANCE_AUTO

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Automatic input impedance provides the lowest measure noise with the highest isolation on the device under test (DUT). When automatic input impedance is selected, the 100 mV to 10 V voltage ranges have more than 10 GΩ input impedance. For the 100 V and 1000 V ranges, a 10 MΩ input divider is placed across the HI and LO input terminals.

When the input impedance is set to 10 MΩ, the 100 mV to 1000 V ranges have a 10 MΩ input divider across the HI and LO input terminals. The 10 M impedance provides stable measurements when the terminals are open (approximately 100 μV at 1 PLC).

Choosing automatic input impedance is a balance between achieving low DC voltage noise on the 100 mV and 1 V ranges and optimizing measurement noise due to charge injection. The Model DMM7510 is optimized for low noise and charge injection when the DUT has less than 100 KΩ input resistance. When the DUT input impedance is more than 100 K, selecting an input impedance of 10 MΩ optimizes the measurement for lowest noise on the 100 mV and 1 V ranges. You can achieve short-term low noise and low charge injection on the 100 mV and 1 V ranges with autozero off. For the 10 V to 1000 V ranges, both input impedance settings achieve low charge injection.

Example

dmm.measure.func = dmm.FUNC_DC_VOLTAGE dmm.measure.inputimpedance = dmm.IMPEDANCE_AUTO	Set input impedance to be set automatically when the DC voltage function is selected.

Also see

dmm.measure.opendetector (on page 8-169)

dmm.measure.limit[Y].audible

This attribute determines if the instrument beeper sounds when a limit test passes or fails, or disables the beeper.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	Continuity: dmm.AUDIBLE_PASS Other functions: dmm.AUDIBLE_NONE

Usage

```
state = dmm.measure.limit[Y].audible
dmm.measure.limit[Y].audible = state
```

state	When the beeper sounds: • Never: dmm.AUDIBLE_NONE • On test failure: dmm.AUDIBLE_FAIL • On test pass: dmm.AUDIBLE_PASS
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The tone and length of beeper cannot be adjusted.

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

dmm.measure.limit[Y].enable (on page 8-156)

dmm.measure.limit[Y].autoclear

This attribute indicates if the test result for limit *Y* should be cleared automatically or not.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.ON

Usage

```
value = dmm.measure.limit[Y].autoclear
dmm.measure.limit[Y].autoclear = value
```

value	The auto clear setting: • Disable: `dmm.OFF` • Enable: `dmm.ON`
Y	Limit number: `1` or `2`

Functions

`dmm.FUNC_DC_VOLTAGE`	`dmm.FUNC_RESISTANCE`	`dmm.FUNC_ACV_FREQUENCY`
`dmm.FUNC_AC_VOLTAGE`	`dmm.FUNC_4W_RESISTANCE`	`dmm.FUNC_ACV_PERIOD`
`dmm.FUNC_DC_CURRENT`	`dmm.FUNC_DIODE`	`dmm.FUNC_DCV_RATIO`
`dmm.FUNC_AC_CURRENT`	`dmm.FUNC_CAPACITANCE`	`dmm.FUNC_DIGITIZE_CURRENT`
`dmm.FUNC_TEMPERATURE`	`dmm.FUNC_CONTINUITY`	`dmm.FUNC_DIGITIZE_VOLTAGE`

Details

When auto clear is set to on for a measure function, limit conditions are cleared automatically after each measurement. If you are making a series of measurements, the instrument shows the limit test result of the last measurement for the pass or fail indication for the limit.

If you want to know if any of a series of measurements failed the limit, set the auto clear setting to off. When this set to off, a failed indication is not cleared automatically. It remains set until it is cleared with the clear command.

The auto clear setting affects both the high and low limits.

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

dmm.measure.limit[Y].enable (on page 8-156)

dmm.measure.limit[Y].clear()

This function clears the results of the limit test defined by *Y*.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.limit[Y].clear()
```

Y	Limit number: `1` or `2`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Use this command to clear the test results of limit *Y* when the limit auto clear option is turned off. Both the high and low test results are cleared.

To avoid the need to manually clear the test results for a limit, turn the auto clear option on.

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

dmm.measure.limit[Y].autoclear (on page 8-154)

dmm.measure.limit[Y].enable

This attribute enables or disables a limit test on the measurement from the selected measure function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
state = dmm.measure.limit[Y].enable
dmm.measure.limit[Y].enable = state
```

state	Limit Y testing: • Disable: dmm.OFF • Enable: dmm.ON
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command enables or disables a limit test for the selected measurement function. When this attribute is enabled, the limit Y testing occurs on each measurement made by the instrument. Limit Y testing compares the measurements to the high and low limit values. If a measurement falls outside these limits, the test fails.

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

dmm.measure.limit[Y].autoclear (on page 8-154)
dmm.measure.limit[Y].clear() (on page 8-155)
dmm.measure.limit[Y].fail (on page 8-157)
dmm.measure.limit[Y].high.value (on page 8-158)
dmm.measure.limit[Y].low.value (on page 8-159)

dmm.measure.limit[Y].fail

This attribute queries the results of a limit test.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
value = dmm.measure.limit[Y].fail
```

`value`	The results of the limit test for limit `Y`: • `dmm.FAIL_NONE`: Test passed; measurement under or equal to the high limit • `dmm.FAIL_HIGH`: Test failed; measurement exceeded high limit • `dmm.FAIL_LOW`: Test failed; measurement exceeded low limit • `dmm.FAIL_BOTH`: Test failed; measurement exceeded both limits
`Y`	Limit number: `1` or `2`

Functions

`dmm.FUNC_DC_VOLTAGE`	`dmm.FUNC_RESISTANCE`	`dmm.FUNC_ACV_FREQUENCY`
`dmm.FUNC_AC_VOLTAGE`	`dmm.FUNC_4W_RESISTANCE`	`dmm.FUNC_ACV_PERIOD`
`dmm.FUNC_DC_CURRENT`	`dmm.FUNC_DIODE`	`dmm.FUNC_DCV_RATIO`
`dmm.FUNC_AC_CURRENT`	`dmm.FUNC_CAPACITANCE`	`dmm.FUNC_DIGITIZE_CURRENT`
`dmm.FUNC_TEMPERATURE`	`dmm.FUNC_CONTINUITY`	`dmm.FUNC_DIGITIZE_VOLTAGE`

Details

This command queries the result of a limit test for the selected measurement function.

The response message indicates if the limit test passed or how it failed (on the high or low limit).

If autoclear is set to off, reading the results of a limit test does not clear the fail indication of the test. To clear a failure, send the clear command. To automatically clear the results, set auto clear on.

If auto clear is set to on and you are making a series of measurements, the last measurement limit determines the fail indication for the limit. If auto clear is turned off, the results return a test fail if any of one of the readings failed.

To use this attribute, you must set the limit state to on.

If the readings are stored in a reading buffer, you can use the `bufferVar.statuses` command to see the results.

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

bufferVar.statuses (on page 8-37)
dmm.measure.limit[Y].enable (on page 8-156)

dmm.measure.limit[Y].high.value

This attribute specifies the upper limit for a limit test.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1 for most functions; see **Details**

Usage

```
highLimit = dmm.measure.limit[Y].high.value
dmm.measure.limit[Y].high.value = highLimit
```

highLimit	The value of the upper limit (-1e+12 to 1e+12)
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command sets the high limits for the limit tests for the selected measurement function. When limit testing is enabled for this limit, the instrument generates a fail indication when the measurement value is more than this value.

Default is 0.8 for limit 1 when the diode function is selected; 10 when the continuity function is selected. The default for limit 2 for the diode and continuity functions is 1.

Example

See dmm.measure.limit[Y].low.value (on page 8-159) for an example of how to use this command.

Also see

dmm.measure.limit[Y].enable (on page 8-156)
dmm.measure.limit[Y].low.value (on page 8-159)

dmm.measure.limit[Y].low.value

This attribute specifies the lower limit for limit tests.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	-1 for most functions; see **Details**

Usage

```
lowLimit = dmm.measure.limit[Y].low.value
dmm.measure.limit[Y].low.value = lowLimit
```

lowLimit	The low limit value of limit *Y*; the range is −1E+12 to 1E+12
Y	Limit number: 1 or 2

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command sets the lower limit for the limit tests for the selected measurement function. When limit *Y* testing is enabled, this causes a fail indication to occur when the measurement value is less than this value.

Default is 0.3 for limit 1 when the diode function is selected. The default for limit 2 for the diode function is −1.

Example

This example enables limits 1 and 2 for voltage measurements. Limit 1 is checking for readings to be between 3 and 5 V, while limit 2 is checking for the readings to be between 1 and 7 V. The auto clear feature is disabled, so if any reading is outside these limits, the corresponding fail is 1. Therefore, if any one of the fails is 1, analyze the reading buffer data to determine which reading failed the limits.

```
reset()
-- set the instrument to measure voltage
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
-- set the range to 10 V
dmm.measure.range = 10
-- set the nplc to 0.1
dmm.measure.nplc = 0.1
-- disable auto clearing for limit 1
dmm.measure.limit[1].autoclear = dmm.OFF
-- set high limit on 1 to fail if reading exceeds 5 V
dmm.measure.limit[1].high.value = 5
-- set low limit on 1 to fail if reading is less than 3 V
dmm.measure.limit[1].low.value = 3
--- set the beeper to sound if the reading exceeds the limits for limit 1
dmm.measure.limit[1].audible = dmm.AUDIBLE_FAIL
-- enable limit 1 checking for voltage measurements
dmm.measure.limit[1].enable = dmm.ON
-- disable auto clearing for limit 2
dmm.measure.limit[2].autoclear = dmm.OFF
-- set high limit on 2 to fail if reading exceeds 7 V
dmm.measure.limit[2].high.value = 7
-- set low limit on 2 to fail if reading is less than 1 V
dmm.measure.limit[2].low.value = 1
-- enable limit 2 checking for voltage measurements
dmm.measure.limit[2].enable = dmm.ON
-- set the measure count to 50
dmm.measure.count = 50
-- create a reading buffer that can store 100 readings
LimitBuffer = buffer.make(100)
-- make 50 readings and store them in LimitBuffer
dmm.measure.read(LimitBuffer)
-- Check if any of the 50 readings were outside of the limits
print("limit 1 results = " .. dmm.measure.limit[1].fail)
print("limit 2 results = " .. dmm.measure.limit[2].fail)
-- clear limit 1 conditions
dmm.measure.limit[1].clear()
-- clear limit 2 conditions
dmm.measure.limit[2].clear()
```

Example output that shows all readings are within limit values (all readings between 3 V and 5 V):

```
limit 1 results = dmm.FAIL_NONE
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 high values (a 6 V reading would cause this condition or a reading greater than 5 V but less than 7 V):

```
limit 1 results = dmm.FAIL_HIGH
limit 2 results = dmm.FAIL_NONE
```

Example output showing at least one reading failed limit 1 and 2 low values (a 0.5 V reading would cause this condition or a reading less than 1 V):

```
limit 1 results = dmm.FAIL_LOW
limit 2 results = dmm.FAIL_LOW
```

Also see

dmm.measure.limit[Y].autoclear (on page 8-154)
dmm.measure.limit[Y].clear() (on page 8-155)
dmm.measure.limit[Y].enable (on page 8-156)
dmm.measure.limit[Y].fail (on page 8-157)
dmm.measure.limit[Y].high.value (on page 8-158)

dmm.measure.linesync

This attribute determines if line synchronization is used during the measurement.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
state = dmm.measure.linesync
dmm.measure.linesync = state
```

state	Disable line sync: dmm.OFF
	Enable line sync: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When line synchronization is enabled, measurements are initiated at the first positive-going zero crossing of the power line cycle after the trigger.

Example

dmm.measure.func = dmm.FUNC_DC_CURRENT dmm.measure.linesync = dmm.ON	Set line synchronization on for DC current measurements.

Also see

Line cycle synchronization (on page 4-1)

dmm.measure.math.enable

This attribute enables or disables math operations on measurements for the selected measurement function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
value = dmm.measure.math.enable
dmm.measure.math.enable = value
```

value	The math enable setting: • Disable: dmm.OFF • Enable: dmm.ON

Details

When this command is set to on, the math operation specified by the math format command is performed before completing a measurement.

Example

```
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
dmm.measure.math.format = dmm.MATH_PERCENT
dmm.measure.count = 1
dmm.measure.math.percent = dmm.measure.read()
dmm.measure.math.enable = dmm.ON
dmm.measure.count = 5
MathBuffer = buffer.make(100)
dmm.measure.read(MathBuffer)
printbuffer(1, MathBuffer.n, MathBuffer.formattedreadings)
dmm.measure.count = 1
for x = 1, 3 do
    print(dmm.measure.read(MathBuffer))
end
```

Configure the instrument for DC volts and reset the DC volts function to the default settings.
Set math format to percent.
Acquire 1 reading to use as the relative percent value.
Take 5 readings with percent math enabled and store them in a buffer called `MathBuffer` that can store 100 readings.

Take three additional readings without using the reading buffer.

Sample output assuming no load was connected to the instrument:
–100.00242 %, –100.00228 %, –100.00220 %, –100.00233 %, –100.00216 %
–100.00228175
–100.0022889
–100.00210915

Also see

[Calculations that you can apply to measurements](#) (on page 3-6)
[dmm.measure.math.format](#) (on page 8-163)

dmm.measure.math.format

This attribute specifies which math operation is performed on measurements when math operations are enabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.MATH_PERCENT

Usage

```
operation = dmm.measure.math.format
dmm.measure.math.format = operation
```

operation	Math operation to be performed on measurements: • y = mx+b: dmm.MATH_MXB • Percent: dmm.MATH_PERCENT • Reciprocal: dmm.MATH_RECIPROCAL

Details

This specifies which math operation is performed on measurements for the selected measurement function.

You can choose one of the following math operations:

- **y = mx+b**: Manipulate normal display readings by adjusting the m and b factors.
- **Percent**: Displays measurements as the percentage of deviation from a specified reference constant.
- **Reciprocal**: The reciprocal math operation displays measurement values as reciprocals. The displayed value is $1/X$, where X is the measurement value (if relative offset is being used, this is the measured value with relative offset applied).

Math calculations are applied to the input signal after relative offset and before limit tests.

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE`
`dmm.measure.math.format = dmm.MATH_RECIPROCAL`
`dmm.measure.math.enable = dmm.ON`
	Enables the reciprocal math operation on voltage measurements.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.measure.math.enable (on page 8-162)

dmm.measure.math.mxb.bfactor

This attribute specifies the offset, b, for the y = mx + b operation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
offsetFactor = dmm.measure.math.mxb.bfactor
dmm.measure.math.mxb.bfactor = offsetFactor
```

offsetFactor	The offset for the y = mx + b operation; the valid range is −1e12 to +1e12

Details

This attribute specifies the offset (b) for an mx + b operation.

The mx + b math operation lets you manipulate normal display readings (x) mathematically according to the following calculation:

$$y = mx + b$$

Where:

- *y* is the displayed result
- *m* is a user-defined constant for the scale factor
- *x* is the measurement reading (if you are using a relative offset, this is the measurement with relative offset applied)
- *b* is the user-defined constant for the offset factor.

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE`
`dmm.measure.math.format = dmm.MATH_MXB`
`dmm.measure.math.mxb.mfactor = 0.80`
`dmm.measure.math.mxb.bfactor = 50`
`dmm.measure.math.enable = dmm.ON`
	Set the measurement function to voltage.
	Set the scale factor for the mx +b operation to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.measure.math.enable (on page 8-162)
dmm.measure.math.format (on page 8-163)
dmm.measure.math.mxb.mfactor (on page 8-165)

dmm.measure.math.mxb.mfactor

This attribute specifies the scale factor, m, for the y = mx + b math operation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
scaleFactor = dmm.measure.math.mxb.mfactor
dmm.measure.math.mxb.mfactor = scaleFactor
```

scaleFactor	The scale factor; the valid range is −1e12 to +1e12

Details

This command sets the scale factor (m) for an mx + b operation for the selected measurement function.

The mx + b math operation lets you manipulate normal display readings (x) mathematically according to the following calculation:

$$y = mx + b$$

Where:

- *y* is the displayed result
- *m* is a user-defined constant for the scale factor
- *x* is the measurement reading (if you are using a relative offset, this is the measurement with relative offset applied)
- *b* is the user-defined constant for the offset factor

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE`
`dmm.measure.math.format = dmm.MATH_MXB`
`dmm.measure.math.mxb.mfactor = 0.80`
`dmm.measure.math.mxb.bfactor = 50`
`dmm.measure.math.enable = dmm.ON`
	Set the measurement function to voltage.
	Set the scale factor for the mx +b operation to 0.80.
	Set the offset factor to 50.
	Enable the math function.

Also see

dmm.measure.math.percent

This attribute specifies the reference constant that is used when math operations are set to percent.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
reference = dmm.measure.math.percent
dmm.measure.math.percent = reference
```

reference	The reference used when the math operation is set to percent; the range is -1e12 to +1e12

Details

The percent math function displays measurements as percent deviation from a specified reference constant. The percent calculation is:

$$Percent = \left(\frac{input - reference}{reference} \right) \times 100\%$$

Where:

- *Percent* is the result
- *Input* is the measurement (if relative offset is being used, this is the relative offset value)
- *Reference* is the user-specified constant

Example

```dmm.measure.func = dmm.FUNC_DC_VOLTAGE```
```dmm.measure.math.format = dmm.MATH_PERCENT```
```dmm.measure.math.percent = 50```
```dmm.measure.math.enable = dmm.ON```
	Set the measurement function to voltage.
	Set the math operations to percent.
	Set the reference constant to 50 for voltage measurements.
	Enable math operations.

Also see

Calculations that you can apply to measurements (on page 3-6)
dmm.measure.math.enable (on page 8-162)
dmm.measure.math.format (on page 8-163)

dmm.measure.nplc

This command sets the time that the input signal is measured for the selected function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1

Usage

```
nplc = dmm.measure.nplc
dmm.measure.nplc = nplc
```

nplc	The number of power line cycles: • 60 Hz: 0.0005 to 15 • 50 Hz: 0.0005 to 12

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command sets the amount of time that the input signal is measured.

The amount of time is specified as the number of power line cycles (NPLCs). Each PLC for 60 Hz is 16.67 ms (1/60) and each PLC for 50 Hz is 20 ms (1/50). For 60 Hz, if you set the NPLC to 0.1, the measure time is 1.667 ms.

The shortest amount of time results in the fastest reading rate, but increases the reading noise and decreases the number of usable digits.

The longest amount of time provides the lowest reading noise and more usable digits, but has the slowest reading rate.

Settings between the fastest and slowest number of PLCs are a compromise between speed and noise.

If you change the PLCs, you may want to adjust the displayed digits to reflect the change in usable digits.

NOTE

The measurement time can also be set as an aperture time. Changing the NPLC value changes the aperture time and changing the aperture time changes the NPLC value.

Example

```dmm.measure.func = dmm.FUNC_DC_VOLTAGE``` ```dmm.measure.nplc = 0.5```	Sets the measurement time to 0.0083 (0.5/60) seconds.

Also see

dmm.measure.aperture (on page 8-125)
dmm.measure.linesync (on page 8-161)
Using aperture or NPLCs to adjust speed and accuracy (on page 4-2)

dmm.measure.offsetcompensation.enable

This attribute enables or disables offset compensation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	4-wire resistance: dmm.OFF Temperature, 3- or 4-wire RTD: dmm.ON

Usage

```
state = dmm.measure.offsetcompensation.enable
dmm.measure.offsetcompensation.enable = state
```

state	Disable offset compensation: dmm.OFF Enable offset compensation: dmm.ON (for 4-wire resistance, not available for ranges more than 1 MΩ)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The voltage offsets caused by the presence of thermoelectric EMFs (V_{EMF}) can adversely affect resistance measurement accuracy. To overcome these offset voltages, you can use offset-compensated ohms.

For 4-wire resistance measurements, when offset compensation is enabled, the measure range is limited to a maximum of 100 kΩ. Offset compensation is automatically enabled when dry circuit is enabled.

For 2-wire resistance measurements, offset compensation is always set to off.

For temperature measurements, offset compensation is only available when the transducer type is set to 3-wire or 4-wire RTD.

Example

| ```
dmm.measure.func = dmm.FUNC_TEMPERATURE
dmm.measure.transducer = dmm.TRANS_FOURRTD
dmm.measure.offsetcompensation.enable = dmm.ON
print(dmm.measure.read())
``` | Sets the measurement function to resistance. Set the instrument for 4-wire RTD and turn offset compensation on. Make a measurement. |
|---|---|

Also see

dmm.measure.drycircuit (on page 8-144)

dmm.measure.opendetector

This attribute determines if the detection of open leads is enabled or disabled.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	4W Res: dmm.OFF Temperature: dmm.ON

Usage

```
state = dmm.measure.opendetector
dmm.measure.opendetector = state
```

state	Disable open lead detector: dmm.OFF
	Enable open lead detector: dmm.ON

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

For temperature measurements, this is only available when the transducer is set to a thermocouple or one of the RTDs.

Long lengths of thermocouple wire can have a large amount of capacitance, which is seen at the input of the DMM. If an intermittent open occurs in the thermocouple circuit, the capacitance can cause an erroneous on-scale reading. The open thermocouple detection circuit, when enabled, applies a 100 μA pulse of current to the thermocouple before the start of each temperature measurement.

Example

`dmm.measure.func = dmm.FUNC_TEMPERATURE`
`dmm.measure.transducer = dmm.TRANS_THERMOCOUPLE`
`dmm.measure.opendetector = dmm.OFF`
	Set the measure function to temperature.
	Set the transducer type to thermocouple.
	Set open lead detection off.

Also see

dmm.measure.transducer (on page 8-192)
Open lead detection (on page 2-103)

dmm.measure.range

This attribute determines the positive full-scale measure range.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	See **Details**

Usage

```
rangeValue = dmm.measure.range
dmm.measure.range = rangeValue
```

rangeValue	See **Details**

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

You can assign any real number using this command. The instrument selects the closest fixed range that is large enough to measure the entered number. For example, for current measurements, if you expect a reading of approximately 9 mA, set the range to 9 mA to select the 10 mA range. When you read this setting, you see the positive full-scale value of the measurement range that the instrument is presently using.

This command is primarily intended to eliminate the time that is required by the instrument to automatically search for a range.

When a range is fixed, any signal greater than the entered range generates an overrange condition. When an overrange condition occurs, the front panel displays "Overflow" and the remote interface returns the instrument returns 9.9e+37 over the bus.

NOTE

When you set a value for the measurement range, the measurement autorange setting is automatically disabled for the selected measurement function.

The range for measure functions defaults to autorange. When you switch from autorange to range, the range is set to the last selected autorange value.

The following table lists the ranges for each function.

If the measurement function is…	The available ranges are…
DC voltage	100 mV, 1 V, 10 V, 100 V, 1000 V
AC voltage	100 mV, 1 V, 10 V, 100 V, 700 V
DC current	10 μA, 100 μA, 1 mA, 10 mA, 100 mA, 1 A, 3 A 10 A available for rear terminals
AC current	1 mA, 10 mA, 100 mA, 1 A, 3 A 10 A available for rear terminals
2-wire resistance	10 Ω, 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ, 1 MΩ, 10 MΩ, 100 MΩ, 1 GΩ
4-wire resistance	1 Ω, 10 Ω, 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ, 1 MΩ, 10 MΩ, 100 MΩ, 1 GΩ
4-wire resistance with offset compensation off and dry circuit off	1 Ω, 10 Ω, 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ, 1 MΩ, 10 MΩ, 100 MΩ, 1 GΩ
4-wire resistance with offset compensation off and dry circuit on	10 Ω, 100 Ω, 1 kΩ, 10 kΩ
4-wire resistance with offset compensation on and dry circuit off	10 Ω, 100 Ω, 1 kΩ, 10 kΩ, 100 kΩ
4-wire resistance with offset compensation on and dry circuit on	10 Ω, 100 Ω, 1 kΩ, 10 kΩ
Continuity	1 kΩ (fixed)
Diode	10 V (fixed)
Capacitance	1 nF, 10 nF, 100 nF, 1 μF, 10 μF, 100 μF, 1 mF
DC voltage ratio	100 mV, 1 V, 10 V, 100 V, 1000 V

Example

```dmm.measure.func = dmm.FUNC_AC_VOLTAGE dmm.measure.range = 90 print(dmm.measure.range)```	Set the range to 90 V, which selects the 100 V range. Output: ```100```

Also see

dmm.measure.autorange (on page 8-128)

dmm.measure.read()

This function makes measurements, places them in a reading buffer, and returns the last reading.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
reading = dmm.measure.read()
reading = dmm.measure.read(bufferName)
```

reading	The last reading of the measurement process
bufferName	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined buffer; if no buffer is defined, it defaults to defbuffer1

Details

This command initiates measurements using the present function setting, stores the readings in a reading buffer, and returns the last reading.

The dmm.measure.count attribute determines how many measurements are performed.

When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

If you define a specific reading buffer, the reading buffer must exist before you make the measurement.

Example

```voltMeasBuffer = buffer.make(10000)```
 ```dmm.measure.func = dmm.FUNC_DC_VOLTAGE```
 ```print(dmm.measure.read(voltMeasBuffer))```
	Create a buffer named voltMeasBuffer. Set the instrument to measure voltage.
	 Make a measurement that is stored in the voltMeasBuffer and is also printed.

Also see

buffer.make() (on page 8-18)
dmm.measure.count (on page 8-140)
Reading buffers (on page 3-12)
trigger.model.load() — SimpleLoop (on page 8-292)

dmm.measure.readwithtime()

This function initiates measurements and returns the last actual measurement and time information in UTC format without using the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
reading, seconds, fractional = dmm.measure.readwithtime()
dmm.measure.readwithtime(bufferName)
```

reading	The last reading of the measurement process
seconds	Seconds in UTC format
fractional	Fractional seconds
bufferName	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined buffer; if no buffer is specified, this parameter defaults to defbuffer1

Details

This command initiates measurements using the present function setting, stores the readings in a reading buffer, and returns the last reading.

The dmm.measure.count attribute determines how many measurements are performed.

When you use a reading buffer with a command or action that makes multiple readings, all readings are available in the reading buffer. However, only the last reading is returned as a reading with the command.

If you define a specific reading buffer, the reading buffer must exist before you make the measurement.

Example

`print(dmm.measure.readwithtime())`	Print the last measurement and time information in UTC format, which will look similar to: `-1.405293589829e-11  1400904629  0.1950935`

Also see

dmm.measure.count (on page 8-140)
trigger.model.load() — SimpleLoop (on page 8-292)

dmm.measure.rel.acquire()

This function acquires a measurement and stores it as the relative offset value.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.measure.rel.acquire()
```

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command triggers the instrument to make a new measurement for the selected function. This measurement is then stored as the new relative offset level.

When you send this command, the instrument does not apply any math, limit test, or filter settings to the measurement, even if they are set. It is a measurement that is made as if these settings are disabled.

If an error event occurs during the measurement or overflow, the relative offset level remains at the last valid setting.

You must change to the function for which you want to acquire a value before sending this command.

The instrument must have relative offset enabled to use the acquired relative offset value.

After executing this command, you can use the dmm.measure.rel.level attribute to see the last relative level value that was acquired or that was set.

Example

dmm.measure.func = dmm.FUNC_DC_VOLTAGE rel_value = dmm.measure.rel.acquire() dmm.measure.rel.enable = dmm.ON print(rel_value)	Acquires a relative offset level value for voltage measurements, turns the relative offset feature on, and outputs the value.

Also see

dmm.measure.rel.enable (on page 8-175)
dmm.measure.rel.level (on page 8-176)

dmm.measure.rel.enable

This attribute enables or disables the application of a relative offset value to the measurement.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.OFF

Usage

```
state = dmm.measure.rel.enable
dmm.measure.rel.enable = state
```

value	The setting: • Enable: dmm.ON • Disable: dmm.OFF

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

When relative measurements are enabled, all subsequent measured readings are offset by the relative offset value. You can enter a relative offset value or have the instrument acquire a relative offset value.

Each returned measured relative reading is the result of the following calculation:

Displayed reading = Actual measured reading - Relative offset value

Example

| ```
dmm.measure.func = dmm.FUNC_AC_CURRENT
dmm.measure.rel.acquire()
dmm.measure.rel.enable = dmm.ON
``` | Enables the relative measurements for AC current and uses the acquire command to set the relative level attribute. |
|---|---|

Also see

dmm.measure.rel.acquire() (on page 8-174)
dmm.measure.rel.level (on page 8-176)
dmm.measure.rel.method (on page 8-178)

dmm.measure.rel.level

This attribute contains the relative offset value.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.measure.rel.level
dmm.measure.rel.level = value
```

value	Relative offset value for measurements; see **Details**

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command specifies the relative offset value that can be applied to new measurements. When relative offset is enabled, all subsequent measured readings are offset by the value that is set for this command.

You can set this value, or have the instrument acquire a value. If the instrument acquires the value, read this setting to return the value that was measured internally.

The ranges for the relative offset values are listed in the following table.

	Minimum	Maximum
DC voltage	-1000	1000
AC voltage	-700	700
DC current (rear terminals selected)	-10	10
DC current (front terminals selected)	-3	3
AC current (rear terminals selected)	-10	10
AC current (front terminals selected)	-3	3
Resistance	-1e+09	1e+09
4-wire resistance	-1e+09	1e+09
Diode	-10	10
Capacitance	-0.001	0.001
Temperature	-3310	3310
Continuity	-1000	1000
Frequency	-1e+06	1e+06
Period	-1	1
DC voltage ratio - Method set to result	-1E+12	1E+12
DC voltage ratio - Method set to parts	-1000	1000
Digitize voltage	-1000	1000
Digitize current (rear terminals selected)	-10	10
Digitize current (front terminals selected)	-3	3

NOTE

If you have math, limits, or filter operations selected, you can set the relative offset value to include the adjustments made by these operations. To include these operations, set `dmm.measure.rel.level` to `dmm.measure.read()`. The adjustments from these operations are not used if you use the `dmm.measure.rel.acquire()` function to set the relative offset level.

Example

``dmm.measure.func = dmm.FUNC_DC_CURRENT
dmm.measure.rel.level = dmm.measure.read()
dmm.measure.rel.enable = dmm.ON```	Set the measure function to DC current. Set the relative offset level to be the reading with any calculations included. Enable the relative offset.

Also see

Relative offset (on page 3-5)
dmm.measure.rel.acquire() (on page 8-174)
dmm.measure.rel.enable (on page 8-175)

dmm.measure.rel.method

This attribute determines if relative offset is applied to the measurements before calculating the DC voltage ratio value.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.METHOD_PARTS

Usage

```
value = dmm.measure.rel.method
dmm.measure.rel.method = value
```

value	The method used: • Do not apply relative offset: `dmm.METHOD_RESULT` • Apply relative offset: `dmm.METHOD_PARTS`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command determines if relative offset is applied to the voltage measurements before the ratio calculation or if the relative offset is applied to the final calculated value.

When the parts methods is selected, the individual readings each have the relative offset value applied before being used to calculate the measurement reading. When parts is selected, the relative offset value is working with smaller ranges, so an error may occur. Reduce the relative offset value if you receive an error. A relative offset is applied to the sense value and then to the input value.

When the results method is selected, the individual readings do not have the relative offset value applied. The relative offset value is applied to the final calculation.

Example

`dmm.measure.func = dmm.FUNC_DCV_RATIO` `dmm.measure.rel.method = dmm.METHOD_PARTS`	Set the measure function to DC voltage ratio. Set the method to apply relative offset before generating the ratio.

Also see

dmm.measure.rel.enable (on page 8-175)
dmm.measure.rel.level (on page 8-176)
Relative offset (on page 3-5)

dmm.measure.rtdalpha

This attribute contains the alpha value of a user-defined RTD.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0.00385055

Usage

```
value = dmm.measure.rtdalpha
dmm.measure.rtdalpha = value
```

value	The RTD alpha value: `0` to `0.01`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This attribute is only valid when:

- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

```dmm.measure.func = dmm.FUNC_TEMPERATURE```
```dmm.measure.transducer = dmm.TRANS_THREERTD```
```dmm.measure.threertd = dmm.RTD_USER```
```dmm.measure.rtdalpha = .00385```
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to User.
	Set the alpha RTD value to 0.00385.

Also see

dmm.measure.fourrtd (on page 8-149)
dmm.measure.threertd (on page 8-188)
dmm.measure.transducer (on page 8-192)

dmm.measure.rtdbeta

This attribute contains the beta value of a user-defined RTD.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0.10863

Usage

```
value = dmm.measure.rtdbeta
dmm.measure.rtdbeta = value
```

value	The RTD beta value: 0 to 1

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This attribute is only valid when:

- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

```dmm.measure.func = dmm.FUNC_TEMPERATURE dmm.measure.transducer = dmm.TRANS_THREERTD dmm.measure.threertd = dmm.RTD_USER dmm.measure.rtdalpha = .00385 dmm.measure.rtdbeta = .3```	Set the measure function to temperature. Set the transducer type to 3-wire RTD. Set the RTD type to User. Set the alpha RTD value to 0.00385.

Also see

dmm.measure.fourrtd (on page 8-149)
dmm.measure.threertd (on page 8-188)
dmm.measure.transducer (on page 8-192)

dmm.measure.rtddelta

This attribute contains the delta value of a user-defined RTD.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	1.4999

Usage

```
range = dmm.measure.rtddelta
dmm.measure.rtddelta = range
```

range	The RTD delta value: 0 to 5

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This attribute is only valid when:

- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example 1

```
dmm.measure.func = dmm.FUNC_TEMPERATURE
dmm.measure.transducer = dmm.TRANS_THREERTD
dmm.measure.threertd = dmm.RTD_USER
dmm.measure.rtddelta = .00385
```

Set the measure function to temperature.
Set the transducer type to 3-wire RTD.
Set the RTD type to User.
Set the delta RTD value to 0.00385.

Also see

dmm.measure.fourrtd (on page 8-149)
dmm.measure.threertd (on page 8-188)
dmm.measure.transducer (on page 8-192)

dmm.measure.rtdzero

This attribute contains the zero value of a user-defined RTD.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	100

Usage

```
value = dmm.measure.rtdzero
dmm.measure.rtdzero = value
```

value	The zero value of the RTD: 0 to 10000

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This attribute is only valid when:

- The function is set to temperature.
- The transducer type is set to 3 or 4-wire RTD.
- The RTD type is set to user-defined.

Example

```dmm.measure.func = dmm.FUNC_TEMPERATURE```
```dmm.measure.transducer = dmm.TRANS_THREERTD```
```dmm.measure.threertd = dmm.RTD_USER```
```dmm.measure.rtdalpha = 0.00385```
```dmm.measure.rtdzero = 120```
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to User.
	Set the alpha RTD value to 0.00385.
	Set the zero RTD value to 120.

Also see

dmm.measure.threertd (on page 8-188)
dmm.measure.transducer (on page 8-192)

dmm.measure.sense.autorange

This attribute determines if the sense range is set manually or automatically.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.ON

Usage

```
value = dmm.measure.sense.autorange
dmm.measure.sense.autorange = value
```

value	Enable autorange: dmm.ON Disable autorange: dmm.OFF

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This selects whether the range for the denominator of the ratio is selected manually or automatically.

This command determines how the range is selected.

When this command is set to off, you must set the range. If you do not set the range, the instrument remains at the range that was selected by autorange.

When this command is set to on, the instrument automatically goes to the most sensitive range to perform the measurement.

If a range is manually selected through the front panel or a remote command, this command is automatically set to off.

Auto range selects the best range in which to measure the signal that is applied to the input terminals of the instrument. When auto range is enabled, the range increases at 120 % of range and decreases occurs when the reading is <10 % of nominal range. For example, if you are on the 1 volt range and auto range is enabled, the instrument auto ranges up to the 10 volt range when the measurement exceeds 1.2 volts. It auto ranges down to the 100 mV range when the measurement falls below 1 volt.

Example

dmm.measure.func = dmm.FUNC_DCV_RATIO dmm.measure.sense.autorange = dmm.OFF	Select the DC voltage ratio function. Set the sense range to be set manually.

Also see

dmm.measure.sense.range

This attribute determines the positive full-scale range for the sense measurement.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	10

Usage

```
value = dmm.measure.sense.range
dmm.measure.sense.range = value
```

value	Range in volts: `0.1`, `1`, or `10`

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

Determines the full-scale input for the reference measurement in the denominator of the ratio. It also affects the accuracy of the measurements and the maximum signal that can be measured. Autorange is automatically set to off if a specific value is set.

When you assign a range value, the instrument is set on a fixed range that is large enough to measure the assigned value. The instrument selects the best range for measuring the maximum expected value.

For example, if you expect a sense reading of approximately 9 V, set the range to 9 V to select the 10 V range.

This command is primarily intended to eliminate the time that is required by the instrument to select an automatic range.

Note that when you select a fixed range, an overflow condition can occur.

When you read this setting, you see the positive full-scale value of the sense range that the instrument is presently using.

Example

`dmm.measure.func = dmm.FUNC_DCV_RATIO` `dmm.measure.sense.range = 1`	Select the DC voltage ratio function. Set the sense range to 1 V.

Also see

Ranges (on page 3-4)
dmm.measure.sense.autorange (on page 8-183)

dmm.measure.simreftemperature

This attribute sets the simulated reference temperature of the thermocouple reference junction.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	Celsius: 23 Kelvin: 296.15 Fahrenheit: 73.4

Usage

```
value = dmm.measure.simreftemperature
dmm.measure.simreftemperature = value
```

value	The simulated reference temperature: • Celsius: 0 to 65 • Kelvin: 273 to 338 • Fahrenheit: 32 to 149

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This attribute applies to the temperature function when the transducer type is set to thermocouple and the reference junction is set to simulated. It allows you to set the simulated temperature value.

Example

```dmm.measure.func = dmm.FUNC_TEMPERATURE```
```dmm.measure.transducer = dmm.TRANS_THERMOCOUPLE```
```dmm.measure.units = dmm.measure.UNITS_CELSIUS```
```dmm.measure.simreftemperature = 30```
	Sets 30 degrees Celsius as the simulated reference temperature for thermocouples.

Also see

dmm.measure.transducer (on page 8-192)
Temperature measurements (on page 2-117)

dmm.measure.thermistor

This attribute describes the type of thermistor.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.THERM_5000

Usage

```
value = dmm.measure.thermistor
dmm.measure.thermistor = value
```

value	The thermistor type in ohms:
	• 2252: dmm.THERM_2252
	• 5000: dmm.THERM_5000
	• 10000: dmm.THERM_10000

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command is only applicable when the transducer type is set to thermistor.

Example

```dmm.measure.func = dmm.FUNC_TEMPERATURE``` ```dmm.measure.transducer = dmm.TRANS_THERMISTOR``` ```dmm.measure.thermistor = dmm.THERM_2252```	Set measurement function to temperature. Set the transducer type to thermistor. Set the thermistor type to 2252.

Also see

dmm.measure.transducer (on page 8-192)
Temperature measurements (on page 2-117)

dmm.measure.thermocouple

This attribute indicates the thermocouple type.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.THERMOCOUPLE_K

Usage

```
value = dmm.measure.thermocouple
dmm.measure.thermocouple = value
```

value	The thermocouple type: • dmm.THERMOCOUPLE_B • dmm.THERMOCOUPLE_E • dmm.THERMOCOUPLE_J • dmm.THERMOCOUPLE_K • dmm.THERMOCOUPLE_N • dmm.THERMOCOUPLE_R • dmm.THERMOCOUPLE_S • dmm.THERMOCOUPLE_T

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command is only applicable when the transducer type is set to thermocouple.

Example

```dmm.measure.func = dmm.FUNC_TEMPERATURE```
```dmm.measure.transducer = dmm.TRANS_THERMOCOUPLE```
```dmm.measure.thermocouple = dmm.THERMOCOUPLE_J```
	Set the measure function to temperature.
	Set the transducer type to thermocouple.
	Set the thermocouple type to J.

Also see

dmm.measure.transducer (on page 8-192)
dmm.measure.simreftemperature (on page 8-185)
Temperature measurements (on page 2-117)

dmm.measure.threertd

This attribute defines the type of three-wire RTD that is being used.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	PT100

Usage

```
value = dmm.measure.threertd
dmm.measure.threertd = value
```

value	The type for 3-wire RTD: • PT100: `dmm.RTD_PT100` • PT385: `dmm.RTD_PT385` • PT3916: `dmm.RTD_PT3916` • D100: `dmm.RTD_D100` • F100: `dmm.RTD_F100` • User-specified type: `dmm.RTD_USER`

Functions

`dmm.FUNC_DC_VOLTAGE`	`dmm.FUNC_RESISTANCE`	`dmm.FUNC_ACV_FREQUENCY`
`dmm.FUNC_AC_VOLTAGE`	`dmm.FUNC_4W_RESISTANCE`	`dmm.FUNC_ACV_PERIOD`
`dmm.FUNC_DC_CURRENT`	`dmm.FUNC_DIODE`	`dmm.FUNC_DCV_RATIO`
`dmm.FUNC_AC_CURRENT`	`dmm.FUNC_CAPACITANCE`	`dmm.FUNC_DIGITIZE_CURRENT`
`dmm.FUNC_TEMPERATURE`	`dmm.FUNC_CONTINUITY`	`dmm.FUNC_DIGITIZE_VOLTAGE`

Details

The transducer type must be set to temperature and the transducer must be set to 3-wire RTD before you can set the RTD type.

Example

`dmm.measure.func = dmm.FUNC_TEMPERATURE`
`dmm.measure.transducer = dmm.TRANS_THREERTD`
`dmm.measure.threertd = dmm.RTD_D100`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to D100.

Also see

dmm.measure.rtdalpha (on page 8-179)
dmm.measure.rtdbeta (on page 8-180)
dmm.measure.rtddelta (on page 8-181)
dmm.measure.rtdzero (on page 8-182)
dmm.measure.transducer (on page 8-192)
Temperature measurements (on page 2-117)

dmm.measure.threshold.autorange

This attribute determines if the threshold range is set manually or automatically.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.ON

Usage

```
state = dmm.measure.threshold.autorange
dmm.measure.threshold.autorange = state
```

state	The auto range setting: • Disable: `dmm.OFF` • Enable: `dmm.ON`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

This command determines how the range is selected.

When this command is set to off, you must set the range. If you do not set the range, the instrument remains at the range that was selected by autorange.

When this command is set to on, the instrument uses the signal to determine the most sensitive range on which to perform the measurement. The instrument sets the range when a measurement is requested. To set the range, the instrument makes a measurement to determine the range before making the final measurement, which can result in slower reading times. Turn autorange off and set a specific range to increase measure time.

If a range is manually selected through the front panel or a remote command, this command is automatically set to off.

Example

`dmm.measure.func = dmm.FUNC_ACV_PERIOD` `dmm.measure.threshold.autorange = dmm.ON`	Set the measure function to period. Set the threshold autorange on.

Also see

[dmm.measure.threshold.level](#) (on page 8-190)
[dmm.measure.threshold.range](#) (on page 8-191)

dmm.measure.threshold.level

This attribute determines the signal level where the instrument makes frequency or period measurements.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	0

Usage

```
value = dmm.measure.threshold.level
dmm.measure.threshold.level = value
```

value	The level: −700 V to 700 V; dependent on range

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

You need to set an appropriate voltage trigger level in order for the frequency counter to operate properly. The frequency counter only counts cycles when the signal amplitude reaches the trigger level. For example, if you set the trigger level for 10 V, any cycles with peak amplitude less than 10 V are not counted.

You must select a specific threshold range (autorange must be set to off) before setting a level that is not zero.

Example

`dmm.measure.func = dmm.FUNC_ACV_FREQUENCY`
`dmm.measure.threshold.range = 10`
`dmm.measure.threshold.level = 5`
	Set the measure function to frequency.
	Set the threshold range to 10 V.
	Set the threshold level to 5 V.

Also see

dmm.measure.threshold.autorange (on page 8-189)
dmm.measure.threshold.range (on page 8-191)

dmm.measure.threshold.range

This attribute indicates the expected input level of the voltage signal.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	10 V

Usage

```
value = dmm.measure.threshold.range
dmm.measure.threshold.range = value
```

value	The range: 0.1 to 700; instrument selects nearest valid range (100 mV, 1 V, 10 V, 100 V, 700 V)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The range setting conditions the signal. The instrument automatically selects the most sensitive threshold range for the value you enter. For example, if you specify the expected input voltage to be 90 mV, the instrument automatically selects the 100 mV threshold range.

Example

dmm.measure.threshold.range = 50	Set the threshold range for the selected function to 100 V.

Also see

[dmm.measure.threshold.autorange](#) (on page 8-189)
[dmm.measure.threshold.level](#) (on page 8-190)

dmm.measure.transducer

This attribute sets the transducer type for the temperature measurement function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	dmm.TRANS_THERMOCOUPLE

Usage

```
type = dmm.measure.transducer
dmm.measure.transducer = type
```

type	The transducer type: • Thermocouple: `dmm.TRANS_THERMOCOUPLE` • Thermistor: `dmm.TRANS_THERMISTOR` • 3-wire RTD: `dmm.TRANS_THREERTD` • 4-wire RTD: `dmm.TRANS_FOURRTD`

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The transducer type determines the type of temperature measurement that is made. Each transducer type has related settings that must also be set. For example, thermocouple measurements are only made if the type is set is set to thermocouple. You also need to set the thermocouple type when setting up a thermocouple. For a transducer type of four-wire RTD, you also set the RTD type.

Example

`dmm.measure.func = dmm.FUNC_TEMPERATURE`
`dmm.measure.transducer = dmm.TRANS_THREERTD`
`dmm.measure.threertd = dmm.RTD_D100`
	Set the measure function to temperature.
	Set the transducer type to 3-wire RTD.
	Set the RTD type to PT3916.

Also see

dmm.measure.fourrtd (on page 8-149)
dmm.measure.thermistor (on page 8-186)
dmm.measure.thermocouple (on page 8-187)
dmm.measure.threertd (on page 8-188)
Temperature measurements (on page 2-117)

dmm.measure.unit

This attribute sets the units of measurement that are displayed on the front panel of the instrument and stored in the reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list	Configuration script Measure configuration list	Temperature: dmm.UNIT_CELSIUS Voltage: dmm.UNIT_VOLT

Usage

```
value = dmm.measure.unit
dmm.measure.unit = value
```

value	For DC volts and AC volts, select from the following units: • dmm.UNIT_VOLT • dmm.UNIT_DB For temperature, select from the following units: • dmm.UNIT_CELSIUS • dmm.UNIT_KELVIN • dmm.UNIT_FAHRENHEIT

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

The change in measurement units is displayed when the next measurement occurs. You can only change the units for the voltage, temperature, and digitize voltage functions. Other functions have a fixed units setting that cannot be changed.

Example

`dmm.measure.func = dmm.FUNC_DC_VOLTAGE` `dmm.measure.unit = dmm.UNIT_DB`	Changes the front-panel display and buffer readings for voltage measurements to be displayed as decibel readings.

Also see

None

dmm.measure.userdelay[N]

This attribute sets a user-defined delay that you can use in the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Measure configuration list Function change	Configuration script Measure configuration list	0

Usage

```
delayTime = dmm.measure.userdelay[N]
dmm.measure.userdelay[N] = delayTime
```

delayTime	The delay (0 for no delay, or 167 ns to 10,000 s)
N	The user delay to which this time applies (1 to 5)

Functions

dmm.FUNC_DC_VOLTAGE	dmm.FUNC_RESISTANCE	dmm.FUNC_ACV_FREQUENCY
dmm.FUNC_AC_VOLTAGE	dmm.FUNC_4W_RESISTANCE	dmm.FUNC_ACV_PERIOD
dmm.FUNC_DC_CURRENT	dmm.FUNC_DIODE	dmm.FUNC_DCV_RATIO
dmm.FUNC_AC_CURRENT	dmm.FUNC_CAPACITANCE	dmm.FUNC_DIGITIZE_CURRENT
dmm.FUNC_TEMPERATURE	dmm.FUNC_CONTINUITY	dmm.FUNC_DIGITIZE_VOLTAGE

Details

To use this command in a trigger model, assign the delay to the dynamic delay block.

The delay is specific to the selected function.

Example

```
dmm.measure.userdelay[1] = 5
trigger.model.setblock(1, trigger.BLOCK_DELAY_DYNAMIC, trigger.USER_DELAY_M1)
trigger.model.setblock(2, trigger.BLOCK_MEASURE)
trigger.model.setblock(3, trigger.BLOCK_BRANCH_COUNTER, 10, 1)
trigger.model.initiate()
```

Set user delay 1 for measurements to 5 s.
Set trigger block 1 to a dynamic delay that calls user delay 1.
Set trigger block 2 to make a measurement.
Set trigger block 3 to branch to block 1 ten times.
Start the trigger model.

Also see

dmm.reset()

This function resets commands that begin with `dmm.` to their default settings.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
dmm.reset()
```

Example

`dmm.reset()`	Resets the DMM commands to their default settings.

Also see

[reset()](#) (on page 8-233)

dmm.terminals

This attribute describes which set of input and output terminals the instrument is using.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
terminals = dmm.terminals
```

`terminals`	Using the front-panel input and output terminals: `dmm.TERMINALS_FRONT` Using the rear-panel input and output terminals: `dmm.TERMINALS_REAR`

Details

You must use the front-panel TERMINALS button to change which set of terminals the instrument reads.

Example

`print(dmm.terminals)`	Request information on which terminals are used. Output if the instrument is using the front terminals: `dmm.TERMINALS_FRONT`

Also see

None

dmm.trigger.digitize.stimulus

This attribute sets the instrument to digitize a measurement the next time it detects the specified trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	dmm.EVENT_NONE

Usage

```
event = dmm.trigger.digitize.stimulus
dmm.trigger.digitize.stimulus = event
```

event	The event to use as a stimulus; see **Details**

Details

This command is intended to provide the lowest possible latency between a trigger event such as digital I/O and a reading. It forces the instrument to make a digitize measurement the next time it detects the specified trigger event. Options for the trigger event parameter are listed in the following table.

A digitize function must be active before sending this command. The measurement is digitized for the active function. If a measure function is active, an error is generated.

Before using this command, set the active reading buffer. Readings are stored in the active reading buffer.

If the count is set to more than 1, the first reading is initialized by this trigger. Subsequent readings occur as rapidly as the instrument can make them. If a trigger occurs during the group measurement, the trigger is latched and another group of measurements with the same count will be triggered after the current group completes.

If the stimulus is set to none, this command has no effect on readings.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`

Trigger events	
Event description	**Event constant**
Trigger event blender *N* (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer *N* (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

```
reset()
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.analogtrigger.mode = dmm.MODE_EDGE
dmm.digitize.analogtrigger.edge.slope = dmm.SLOPE_RISING
dmm.digitize.analogtrigger.edge.level = 0.5
dmm.trigger.digitize.stimulus = trigger.EVENT_ANALOGTRIGGER
```

Reset the instrument.
Set the function to digitize voltage.
Set the analog trigger mode to edge.
Set the slope to rising.
Set the level to 0.5 V.
Set the stimulus to be the analog trigger.

Also see

dmm.digitize.count (on page 8-79)
dmm.digitize.func (on page 8-85)

dmm.trigger.measure.stimulus

This attribute sets the instrument to make a measurement the next time it detects the specified trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	dmm.EVENT_NONE

Usage

```
event = dmm.trigger.measure.stimulus
dmm.trigger.measure.stimulus = event
```

event	The event to use as a stimulus; see **Details**

Details

This command is intended to provide the lowest possible latency between an event such as digital I/O and a reading. It forces the instrument to make a measurement the next time it detects the specified trigger event. Options for the trigger event parameter are listed in the following table.

A measure function must be active before sending this command. The measurement is made for the active measure function. If a digitize function is active, an error is generated.

Before using this command, set the active reading buffer. Readings are stored in the active reading buffer.

If the count is set to more than 1, the first reading is initialized by this trigger. Subsequent readings occur as rapidly as the instrument can make them. If a trigger occurs during the group measurement, the trigger is latched and another group of measurements with the same count will be triggered after the current group completes.

If the stimulus is set to none, this command has no effect on readings.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`

Trigger events	
Event description	**Event constant**
Trigger event blender *N* (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer *N* (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

`reset()`
`dmm.measure.func = dmm.FUNC_DC_CURRENT`
`dmm.trigger.measure.stimulus =`
`    trigger.EVENT_DISPLAY`
`dmm.measure.count = 10`
	Reset the instrument.
	Set the function to DC current.
	Set the trigger stimulus to be the front-panel TRIGGER key.
	Set the count to 10.
	Press the TRIGGER key to generate 10 readings that are stored in the active reading buffer.

Also see

None

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 removes all events from the event log, including entries in the front-panel event log.

Also see

eventlog.next() (on page 8-201)
eventlog.save() (on page 8-204)
Using the event log (on page 2-152)

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; set a cumulative integer value that represents the event log types to: • Errors only: `eventlog.SEV_ERROR` or 1 • Warnings only: `eventlog.SEV_WARN` or 2 • Errors and warnings only: `eventlog.SEV_WARN｜eventlog.SEV_ERROR` or 3 • Information only: `eventlog.SEV_INFO` or 4 • Errors and information only: `eventlog.SEV_INFO｜eventlog.SEV_ERROR` or 5 • Warnings and information only: `eventlog.SEV_INFO｜eventlog.SEV_WARN` or 6 • All events: `eventlog.SEV_ALL` or 7

Details

A count finds the number of unread events in the event log. You can specify the event types to return, or return the count for all events.

This command reports the number of events that have occurred since the command was last sent and or since the event log was last cleared.

Events are read automatically when `localnode.showevents` is enabled. You can also read them individually with `eventlog.next()`.

Example

`print(eventlog.getcount(eventlog.SEV_INFO))`	Displays the present number of unread information messages in the instrument event log. If there are three information messages in the event log, output is: 3

Also see

eventlog.next()

This function returns the oldest unread event message from the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
eventNumber, message, severity, nodeID, timeSeconds, timeNanoSeconds =
   eventlog.next()
eventNumber, message, severity, nodeID, timeSeconds, timeNanoSeconds =
   eventlog.next(eventType)
```

eventNumber	The event number
message	A description of the event
severity	The severity of the event, • Error: 1 • Warning: 2 • Information: 4
nodeID	The TSP-Link node where the event occurred or 0 if the event occurred on the local node
timeSeconds	The seconds portion of the time when the event occurred
timeNanoSeconds	The fractional seconds portion of the time when the event occurred
eventType	Limits the return to specific event log types; set a cumulative integer value that represents the event log types to: • Errors only: eventlog.SEV_ERROR or 1 • Warnings only: eventlog.SEV_WARN or 2 • Errors and warnings only: eventlog.SEV_WARN|eventlog.SEV_ERROR or 3 • Information only: eventlog.SEV_INFO or 4 • Errors and information only: eventlog.SEV_INFO|eventlog.SEV_ERROR or 5 • Warnings and information only: eventlog.SEV_INFO|eventlog.SEV_WARN or 6 • All events: eventlog.SEV_ALL or 7

Details

When an event occurs on the instrument, it is placed in the event log. The `eventlog.next()` command retrieves an unread event from the event log. Once an event is read, it can no longer be accessed remotely. However, it can be viewed on the front panel. When `localnode.showevents` is enabled, this command never returns an event because those events are automatically read and sent to the remote interface.

To read multiple events, execute this command multiple times.

If there are no entries in the event log, the following is returned:

`0  No error     0     0     0     0`

If the event type is not defined, an event of any type is returned.

Example

```
print(eventlog.next(5))
```

Get the oldest error or information event from the event log.
Example output:
```
-285 TSP Syntax error at line 1: unexpected symbol near `0' 1 0 1367806152
   652040060
```

Also see

eventlog.clear() (on page 8-199)
eventlog.getcount() (on page 8-200)
eventlog.save() (on page 8-204)
Using the event log (on page 2-152)

eventlog.post()

This function allows you to post your 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, eventType)
```

message	String that contains the message
eventType	The type of event; if no event is defined, defaults to eventlog.SEV_INFO: • Error: eventlog.SEV_ERROR or 1 • Warning: eventlog.SEV_WARN or 2 • Information: eventlog.SEV_INFO or 4

Details

You can use this command to create your own event log entries and assign a severity level to them. This can be useful for debugging and status reporting.

From the front panel, you must set the Log Warnings and Log Information options on to have the custom warning and information events placed into the event log.

Example

eventlog.clear() eventlog.post("my error", eventlog.SEV_ERROR) print(eventlog.next())	Posts an event named "my error". Output: 1005 User: my error 1 0 1359414094 769632040

Also see

[Using the event log](#) (on page 2-152)

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	Limits the return to specific event log types; set a cumulative integer value that represents the event log types to: • Errors only: `eventlog.SEV_ERROR` or 1 • Warnings only: `eventlog.SEV_WARN` or 2 • Errors and warnings only: `eventlog.SEV_WARN|eventlog.SEV_ERROR` or 3 • Information only: `eventlog.SEV_INFO` or 4 • Errors and information only: `eventlog.SEV_INFO|eventlog.SEV_ERROR` or 5 • Warnings and information only: `eventlog.SEV_INFO|eventlog.SEV_WARN` or 6 • All events: `eventlog.SEV_ALL` or 7 (default)

Details

This command saves all event log entries to a USB flash drive.

If you do not define an event type, the instrument saves all event log entries.

The extension `.csv` is automatically added to the file name.

Example

`eventlog.save("/usb1/WarningsApril", eventlog.SEV_WARN)`	Save warning messages to a `.csv` file on a USB flash drive

Also see

[eventlog.next()](#) (on page 8-201)

exit()

This function stops a script that is presently running.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
exit()
```

Details

Terminates script execution when called from a script that is being executed.

This command does not wait for overlapped commands to complete before terminating script execution. If overlapped commands are required to finish, use the `waitcomplete()` function before calling `exit()`.

Also see

file.close()

This function closes a file on the USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
file.close(fileNumber)
```

fileNumber	The file number returned from the `file.open()` function to close

Details

Note that files are automatically closed when the file descriptors are garbage collected.

Example

```
file_num = file.open("/usb1/GENTRIGGER", file.MODE_WRITE)
file.close(file_num)
```
Open the file `GENTRIGGER` for writing, then close it.

Also see

file.flush()

This function writes buffered data to a file on the USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
file.flush(fileNumber)
```

fileNumber	The file number returned from the `file.open()` function to flush

Details

The `file.write()` function may buffer data instead of writing immediately to the USB flash drive. Use `file.flush()` to flush this 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, flush the file to prevent loss of data because of an aborted test.

Also see

None

file.mkdir()

This function creates a directory at the specified path on the USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
file.mkdir(path)
```

path	The path of the directory

Details

The directory path must be absolute.

Example

`file.mkdir("TestData")`	Create a new directory named `TestData`.

Also see

None

file.open()

This function opens a file on the USB flash drive for later reference.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
fileNumber = file.open(fileName, accessType)
```

fileNumber	A number identifying the open file that you use with other file commands to write, read, flush, or close the file after opening
fileName	The file name to open, including the full path of file
accessType	The type of action to do: • Append the file: `file.MODE_APPEND` • Read the file: `file.MODE_READ` • Write to the file: `file.MODE_WRITE`

Details

The path to the file to open must be absolute.

The root folder of the USB flash drive has the following absolute path:

`"/usb1/"`

Example

`file_num = file.open("/usb1/testfile.txt",`
`    file.MODE_WRITE)`
`if file_num != nil then`
`    file.write(file_num, "This is my test file")`
`    file.close(file_num)`
`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

None

file.read()

This function reads data from a file on the USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

fileContents = file.read(*fileNumber*, *readAction*)

fileContents	The contents of the file based on the *readAction* parameter
fileNumber	The file number returned from the file.open() function to read
readAction	The action: • Return the next line; returns nil if the present file position is at the end of the file: file.READ_LINE • Return a string that represents the number found; returns an event string if no number was found; returns nil if the current file position is at the end of file: file.READ_NUMBER • Return the whole file, starting at the present position; returns nil if the present file position is at the end of the file: file.READ_ALL

Details

This command reads data from a file.

Example

``` file_num = file.open("/usb1/testfile.txt", file.MODE_READ) if file_num != nil then file_contents = file.read(file_num, file.READ_ALL) file.close(file_num) end ```	Open testfile.txt on the USB flash drive for reading. If it opens successfully, read the entire contents of the file and store it in variable file_contents. Close the file.

Also see

None

file.usbdriveexists()

This function detects if a USB flash drive is inserted into the front-panel USB port.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

> *driveInserted* = file.usbdriveexists()
>
> | *driveInserted* | 0: No flash drive is detected |
> |---|---|
> | | 1: Flash drive is detected |

Details

You can call this command from a script to verify that a USB flash drive is inserted before attempting to write data to it.

Example

> | print(file.usbdriveexists()) | If the USB flash drive is not inserted in the USB port on the front panel, this returns 0. |
> |---|---|

Also see

None

file.write()

This function writes data to a file on the USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
file.write(fileNumber, string)
```

fileNumber	The file number returned from the `file.open()` function to write
string	The data to write to the file

Details

The `file.write()` function may buffer data; it may not be written to the USB flash drive immediately. Use the `file.flush()` function to immediately write buffered data to the drive.

You must use the `file.close()` command to close the file after writing.

Example

```file_num = file.open("testfile.txt", file.MODE_WRITE) if file_num != nil then file.write(file_num, "This is my test file") file.close(file_num) 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

file.close() (on page 8-205)
file.flush() (on page 8-206)

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	Restore configuration Instrument reset Power cycle	Configuration script	0 (Automatic)

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 from `1` to `16`; set to `0` to have the instrument select the precision automatically based on the number that is being formatted

Details

This attribute specifies the precision (number of digits) for numeric data printed with the `print()`, `printbuffer()`, and `printnumber()` functions. The `format.asciiprecision` attribute is only used with the ASCII format. The precision value must be a number from 0 to 16.

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 = 10 x = 2.54 printnumber(x) format.asciiprecision = 3 printnumber(x) ```	Output: 2.540000000e+00 2.54e+00

Also see

[format.byteorder](#) (on page 8-212)
[format.data](#) (on page 8-213)
[print()](#) (on page 8-228)
[printbuffer()](#) (on page 8-229)
[printnumber()](#) (on page 8-232)

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)	No	Restore configuration Instrument reset Power cycle	Configuration script	format.LITTLEENDIAN

Usage

```
order = format.byteorder
format.byteorder = order
```

order	Byte order value as follows: • Most significant byte first: `format.BIGENDIAN` • Least significant byte first: `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.REAL32` and `format.REAL64` data formats.

If you are sending data to a computer with a Microsoft Windows operating system, select the `format.LITTLEENDIAN` byte order.

Example

```x = 1.23```
```format.data = format.REAL32```
```format.byteorder = format.LITTLEENDIAN```
```printnumber(x)```
```format.byteorder = format.BIGENDIAN```
```printnumber(x)```
	Output depends on the terminal program you use, but will look something like:
	#0¤p??
	#0??p¤

Also see

[format.asciiprecision](#) (on page 8-211)
[format.data](#) (on page 8-213)
[printbuffer()](#) (on page 8-229)
[printnumber()](#) (on page 8-232)

format.data

This attribute sets the data format for data that is printed using the `printnumber()` and `printbuffer()` functions.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Restore configuration Instrument reset Power cycle	Configuration script	format.ASCII

Usage

```
value = format.data
format.data = value
```

value	The format to use for data, set to one of the following values: • ASCII format: `format.ASCII` • Single-precision IEEE Std 754 binary format: `format.REAL32` • Double-precision IEEE Std 754 binary format: `format.REAL64`

Details

You can control the precision of numeric values with the `format.asciiprecision` attribute. If `format.REAL32` or `format.REAL64` is selected, you can select the byte order with the `format.byteorder` attribute.

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.

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
#0ñÔÈ̃Sû!   @

Also see

gpib.address

This attribute contains the GPIB address.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Not applicable	Nonvolatile memory	16

Usage

```
address = gpib.address
gpib.address = address
```

address	The GPIB address of the instrument (1 to 30)

Details

The address can be set to any address value from 1 to 30. However, the address must be unique in the system. It cannot conflict with an address that is assigned to another instrument or to the GPIB controller.

A new GPIB address takes effect when the command to change it is processed. If there are response messages in the output queue when this command is processed, they must be read at the new address.

If command messages are being queued (sent before this command has executed), the new settings may take effect in the middle of a subsequent command message, so care should be exercised when setting this attribute from the GPIB interface.

You should allow ample time for the command to be processed before attempting to communicate with the instrument again. After sending this command, make sure to use the new address to communicate with the instrument.

The `reset()` function does not affect the GPIB address.

Example

``` gpib.address = 26 address = gpib.address print(address) ```	Sets the GPIB address and reads the address. Output: `2.600000e+01`

Also see

[GPIB setup](#) (on page 2-63)

lan.ipconfig()

This function specifies the LAN configuration for the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No	Rear panel 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:
	`lan.MODE_AUTO`: The instrument automatically assigns LAN settings
	`lan.MODE_MANUAL`: You must specify the LAN settings
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.lxidomain

This attribute contains the LXI domain.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	LAN restore defaults	Nonvolatile memory	0

Usage

```
domain = lan.lxidomain
lan.lxidomain = domain
```

domain	The LXI domain number (0 to 255)

Details

This attribute sets the LXI domain number.

All outgoing LXI packets are generated with this domain number. All inbound LXI packets are ignored unless they have this domain number.

Example

print(lan.lxidomain)	Displays the LXI domain.

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 8-215)

localnode.access

This attribute contains the type of access users have to the instrument through different interfaces.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Not applicable	Nonvolatile memory	localnode.ACCESS_FULL

Usage

```
accessType = localnode.access
localnode.access = accessType
```

accessType	The type of access: • Full access for all users from all interfaces: `localnode.ACCESS_FULL` • Allows access by one remote interface at a time with logins required from other interfaces: `localnode.ACCESS_EXCLUSIVE` • Allows access by one remote interface at a time with passwords required on all interfaces: `localnode.ACCESS_PROTECTED` • Allows access by one interface (including the front panel) at a time with passwords required on all interfaces: `localnode.ACCESS_LOCKOUT`

Details

When access is set to full, the instrument accepts commands from any interface with no login or password.

When access is set to exclusive, you must log out of one remote interface and log into another one to change interfaces. You do not need a password with this access.

Protected access is similar to exclusive access, except that you must enter a password when logging in.

When the access is set to locked out, a password is required to change interfaces, including the front panel interface.

Under any access type, if a script is running on one remote interface when a command comes in from another remote interface, the command is ignored and the message "FAILURE: A script is running, use ABORT to stop it" is generated.

Example

`localnode.access = localnode.ACCESS_LOCKOUT`
`login admin`
`logout`
	Set the instrument access to locked out.
	Log into the interface using the default password.
	Log out of the interface.

Also see

[localnode.password](#) (on page 8-220)

localnode.gettime()

This function retrieves the instrument date and time.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
localnode.gettime()
```

Details

The time is returned 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

`print(os.date('%c', gettime()))`	Example output: `Wed Mar 31 14:25:31 2010`

Also see

localnode.settime() (on page 8-223)

localnode.internaltemp

This attribute returns the internal temperature of the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
temperature = localnode.internaltemp
```

`temperature`	The internal temperature of the instrument

Details

Returns the last recorded value of internal temperature of the instrument in Celsius (°C). The instrument checks internal temperature when it updates references when autozero is on. Internal temperature is not checked if autozero is set to off. It can also become stale when digitize measurements are used or when measurements take a long time.

If the temperature changes more than ±5 °C, the instrument logs an event and displays a message on the front panel that recommends that you perform auto calibration.

Example

`print(localnode.internaltemp)`	Returns the internal temperature of the instrument. Example output: `53.732574528`

Also see

acal.lastrun.internaltemp (on page 8-8)
acal.run() (on page 8-12)

localnode.linefreq

This attribute contains the power line frequency setting that is used for NPLC calculations.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Power cycle	Not applicable	Not applicable

Usage

```
frequency = localnode.linefreq
```

frequency	The detected line frequency: 50 or 60

Details

The instrument automatically detects the power line frequency (either 50 Hz or 60 Hz) when the instrument is powered on.

Example

`frequency = localnode.linefreq` `print(frequency)`	Reads line frequency setting.

Also see

None

localnode.model

This attribute stores the model number.

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

Example

`print(localnode.model)`	Outputs the model number of the local node. For example: `DMM7510`

Also see

localnode.serialno (on page 8-222)

localnode.password

This attribute stores the instrument password.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (W)	No	Rear-panel LAN reset	Nonvolatile memory	"admin"

Usage

```
localnode.password = "password"
```

password	A string that contains the instrument password (maximum 30 characters)

Details

When the access to the instrument is set to protected or lockout, this is the password that is used to gain access.

If you forget the password, you can reset the password to the default:

1. On the front panel, press **MENU**.
2. Under System, select **Info/Manage**.
3. Select **Password Reset**.

You can also reset the password and the LAN settings from the rear panel by inserting a straightened paper clip into hole below LAN RESET.

Example

`localnode.password = "N3wpa55w0rd"`	Changes the password to `N3wpa55w0rd`.

Also see

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	Do not generate prompts: `localnode.DISABLE` Generate prompts: `localnode.ENABLE`

Details

When the prompting mode is enabled, the instrument generates prompts when the instrument is ready 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 event log 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 event log when command processing completed.

- **>>>>.** 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 GPIB connection is active, they will not be enabled for a subsequent USB connection.

NOTE
Do not disable prompting when using Test Script Builder. Test Script Builder requires prompts and sets the prompting mode automatically. If you disable prompting, the instrument will stop responding when you communicate using Test Script Builder because it is waiting for a common complete prompt from Test Script Builder.

Example

`localnode.prompts = localnode.ENABLE`	Enable prompting.

Also see

[tsplink.initialize()](#) (on page 8-341)

localnode.prompts4882

This attribute enables and disables the generation of prompts for IEEE Std 488.2 common commands.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Power cycle	Not saved	localnode.ENABLE

Usage

```
prompting = localnode.prompts4882
localnode.prompts4882 = prompting
```

prompting	IEEE Std 488.2 prompting mode: • Disable prompting: `localnode.DISABLE` • Enable prompting: `localnode.ENABLE`

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 will 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 = localnode.DISABLE`	Disables IEEE Std 488.2 common command prompting.

Also see

[localnode.prompts](#) (on page 8-220)

localnode.serialno

This attribute stores the instrument's serial number.

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 indicates the instrument serial number.

Example

```display.clear()```
```display.settext(display.TEXT2, "Serial #: " ..localnode.serialno)```
```display.changescreen(display.SCREEN_USER_SWIPE)```

Clears the instrument display.
Places the serial number of this instrument on the bottom line of the USER swipe screen display. Displays the USER swipe screen.

Also see

[localnode.model](#) (on page 8-219)
[localnode.version](#) (on page 8-225)

localnode.settime()

This function sets the date and time of the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
localnode.settime()
localnode.settime(year, month, day, hour, minute, second)
localnode.settime(hour, minute, second)
localnode.settime(os.time({year, month, day}))
localnode.settime(os.time({year = year, month = month, day = day, hour = hour, min
    = minute, sec = second}))
```

year	Year; must be more than 1970
month	Month (1 to 12)
day	Day (1 to 31)
hour	Hour in 24-hour time format (0 to 23)
minute	Minute (0 to 59)
second	Second (0 to 59)

Details

Internally, the instrument bases time in UTC time. UTC time is specified as the number of seconds since Jan 1, 1970, UTC. You can use UTC time from a local time specification, or you can use UTC time from another source (for example, your computer).

When called without a parameter (the first form), the function returns the current time.

Example 1

`localnode.settime(2014, 3, 31, 14, 25, 0)` `print(localnode.settime())`	Sets the date and time to Mar 31, 2014 at 2:25 pm and verifies the time. Output: `Mon Mar 31 14:25:09 2014`

Example 2

`systemTime = os.time({year = 2014,`
`        month = 3,`
`        day = 31,`
`        hour = 14,`
`        min = 25})`
`localnode.settime(systemTime)`
`print(os.date('%c', gettime()))`
	Sets the date and time to Mar 31, 2014 at 2:25 pm.
	Output:
	`Wed Mar 31 14:25:00 2010`

Also see

[localnode.gettime()](#) (on page 8-218)

localnode.showevents

This attribute sets whether or not the instrument automatically outputs generated events to the remote interface.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Power cycle	Not saved	0 (no events sent)

Usage

```
errorMode = localnode.showevents
localnode.showevents = errorMode
```

errorMode	The errors that are returned: • No events: `0` • Errors only: `1` (`eventlog.SEV_ERROR`) • Warnings only: `2` (`eventlog.SEV_WARN`) • Errors and warnings: `3` (`eventlog.SEV_ERROR`| `eventlog.SEV_WARN`) • Information only: `4` (`eventlog.SEV_INFO`) • Information and errors: `5` (`eventlog.SEV_INFO` |`eventlog.SEV_ERROR`) • Warnings and information: `6` (`eventlog.SEV_INFO`|`eventlog.SEV_WARN`) • All events: `7` (`eventlog.SEV_ALL`)

Details

Enable this attribute to have the instrument output generated events to the remote interface.

Events are output after a command message is executed but before prompts are issued (if prompts are enabled with `localnode.prompts`).

If this attribute is disabled, use `eventlog.next()` to retrieve unread events from the event log.

Events are enabled or disabled only for the remote interface that is active when you send the command. For example, if you enable show events when the GPIB connection is active, they will not be enabled for a subsequent USB connection.

Example

```
localnode.showevents = eventlog.SEV_ERROR | eventlog.SEV_INFO
trigger.digin[3].edge = trigger.EDGE_EITHER
```

Send generated error and warning messages.
Example output if the edge cannot be sent to either:
```
1805, Settings conflict: setting input edge when line 3 set for digital
```

Also see

eventlog.clear() (on page 8-199)
localnode.prompts (on page 8-220)

localnode.version

This attribute stores the firmware version of the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
version = localnode.version
```

version	Instrument version level

Details

This attribute indicates the version number of the firmware that is presently running in the instrument.

Example

`print(localnode.version)`	Outputs the present version level. Example output: `1.0.0a`

Also see

localnode.model (on page 8-219)
localnode.serialno (on page 8-222)

node[N].execute()

This function starts test scripts on a remote TSP-Link node.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes (see **Details**)			

Usage

```
node[N].execute(scriptCode)
```

N	The node number of this instrument (1 to 64)
scriptCode	A string containing the source code

Details

This command is only applicable to TSP-Link systems. You can use this command to use the remote master node to run a script on the specified node. This function does not run test scripts on the master node; only on the subordinate node when initiated by the master node.

This function may only be called when the group number of the node is different than the node of the master.

This function does not wait for the script to finish execution.

Example 1

`node[2].execute(sourcecode)`	Runs script code on node 2. The code is in a string variable called `sourcecode`.

Example 2

`node[3].execute("x = 5")`	Runs script code in string constant (`"x = 5"`) to set `x` equal to 5 on node 3.

Example 3

`node[32].execute(TestDut.source)`	Runs the test script stored in the variable `TestDut` (previously stored on the master node) on node 32.

Also see

tsplink.group (on page 8-340)

node[N].getglobal()

This function returns the value of a global variable.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
value = node[N].getglobal(name)
```

value	The value of the variable
N	The node number of this instrument (1 to 64)
name	The global variable name

Details

This function retrieves the value of a global variable from the run-time environment of this node.

Do not use this command to retrieve the value of a global variable from the local node. Instead, access the global variable directly. This command should only be used from a remote master when controlling this instrument over a TSP-Link® network.

Example

`print(node[5].getglobal("test_val"))`	Retrieves and outputs the value of the global variable named test_val from node 5.

Also see

node[N].setglobal()

This function sets the value of a global variable.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`node[N].setglobal(name, value)`

N	The node number of this instrument (1 to 64)
name	The global variable name to set
value	The value to assign to the variable

Details

From a remote node, use this function to assign the given value to a global variable.

Do not use this command to create or set the value of a global variable from the local node (set the global variable directly instead). This command should only be used from a remote master when controlling this instrument over a TSP-Link®.

Example

`node[3].setglobal("x", 5)`	Sets the global variable x on node 3 to the value of 5.

Also see

opc()

This function sets the operation complete (OPC) bit after all pending commands, including overlapped commands, have been executed.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
opc()
```

Details

This function causes the operation complete bit in the Status Event Status Register to be set when all previously started local overlapped commands are complete.

Note that each node independently sets its operation complete bits in its own status model. Any nodes that are not actively performing overlapped commands set their bits immediately. All remaining nodes set their own bits as they complete their own overlapped commands.

Example

```opc()```
```waitcomplete()```
```print("1")```
	Output:
	1

Also see

*OPC (on page B-6)
Status model (on page C-1)
waitcomplete() (on page 8-362)

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 any other scripting environment, 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: `1.00000e+01` 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

format.asciiprecision (on page 8-211)

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	Name of first table or reading buffer subtable to print; may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer
bufferVar2	Second table or reading buffer subtable to print; may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer
bufferVarN	The last table or reading buffer subtable to print; may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer
`...`	One or more tables or reading buffer subtables separated with commas

Details

If *startIndex* is set to less than 1 or if *endIndex* is more than the size of the index, 9.910000e+37 is returned for each value outside the allowed index and an event is generated.

If overlapped commands use the specified reading buffers and the commands are not complete (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 readings 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 `format.data` attribute controls the format of the response message.

You can use the *bufferVar* attributes that are listed in the following table with the print buffer command. For example, if `testData` is the buffer, you can use `testData.dates` attribute to print the date of each reading in the `testData` buffer.

You can use *bufferVar*.n to retrieve the number of readings in the specified reading buffer.

Attribute	Description
bufferVar.dates	The dates of readings stored in the reading buffer; see bufferVar.dates (on page 8-27)
bufferVar.extravalues	The additional values (such as the sense voltage from a DC voltage ratio measurement); the reading buffer style must be set to full to use this option; see bufferVar.extravalues (on page 8-28)
bufferVar.formattedreadings	The stored readings formatted as they appear on the front-panel display; see bufferVar.formattedreadings (on page 8-30)
bufferVar.fractionalseconds	The fractional portion of the timestamp (in seconds) of when each reading occurred; see bufferVar.fractionalseconds (on page 8-31)
bufferVar.readings	The readings stored in a specified reading buffer; see bufferVar.readings (on page 8-34)
bufferVar.relativetimestamps	The timestamps, in seconds, when each reading occurred relative to the timestamp of reading buffer entry number 1; see bufferVar.relativetimestamps (on page 8-35)
bufferVar.seconds	The nonfractional seconds portion of the timestamp when the reading was stored in UTC format; see bufferVar.seconds (on page 8-36)
bufferVar.statuses	The status values of readings in the reading buffer; see bufferVar.statuses (on page 8-37)
bufferVar.times	The time when the instrument made the readings; see bufferVar.times (on page 8-39)
bufferVar.timestamps	The timestamps of readings stored in the reading buffer; see bufferVar.timestamps (on page 8-40)
bufferVar.units	The unit of measure that is stored with readings in the reading buffer; see bufferVar.units (on page 8-41)

Example 1

```
reset()
dmm.measure.func = dmm.FUNC_DC_CURRENT
testData = buffer.make(200)
format.data = format.ASCII
format.asciiprecision = 6
trigger.model.load("SimpleLoop", 6, 0, testData)
trigger.model.initiate()
waitcomplete()
printbuffer(1, testData.n, testData.readings, testData.units,
    testData.relativetimestamps)
```

Reset the instrument.

Set the measure function to DC current.

Set the data format and ASCII precision.

Use trigger model SimpleLoop to create a 6 count loop with no delays that stores data in the reading buffer `testBuffer`.

Start the trigger model, wait for the commands to complete, and output the readings.

Use of `testData.n` (`bufferVar`.n) indicates that the instrument should output all readings in the reading buffer. In this example, `testBuffer.n` equals 6.

Example of output data:

```
1.10458e-11, Amp DC, 0.00000e+00, 1.19908e-11, Amp DC, 1.01858e-01, 1.19908e-11, Amp DC,
2.03718e-01, 1.20325e-11, Amp DC, 3.05581e-01, 1.20603e-11, Amp DC, 4.07440e-01, 1.20325e-
11, Amp DC, 5.09299e-01
```

Example 2

```
for x = 1, testData.n do
printbuffer(x,x,testData, testData.units, testData.relativetimestamps)
end
```

Using the same buffer created in Example 1, output the readings, units and relative timestamps on a separate line for each reading.

```
1.10458e-11, Amp DC, 0.00000e+00
1.19908e-11, Amp DC, 1.01858e-01
1.19908e-11, Amp DC, 2.03718e-01
1.20325e-11, Amp DC, 3.05581e-01
1.20603e-11, Amp DC, 4.07440e-01
1.20325e-11, Amp DC, 5.09299e-01
```

Also see

bufferVar.n (on page 8-33)
bufferVar.readings (on page 8-34)
format.asciiprecision (on page 8-211)
format.byteorder (on page 8-212)
format.data (on page 8-213)
printnumber() (on page 8-232)

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.<br>Read the value of x based on these settings.<br>Change the ASCII precision to 3.<br>View how the change affects the output of x and some numbers.<br>Output:<br>`2.540000000e+00`<br>`2.54e+00, 2.54e+00, 3.10e+00` |

Also see

format.asciiprecision (on page 8-211)
format.byteorder (on page 8-212)
format.data (on page 8-213)
print() (on page 8-228)
printbuffer() (on page 8-229)

reset()

This function resets commands to their default settings and clears the buffers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
reset()
reset(system)
```

system	If the node is the master, the entire system is reset: true
	Only the local group is reset: false

Details

The `reset()` command in its simplest form resets the entire TSP-enabled system, including the controlling node and all subordinate nodes.

If you want to reset a specific instrument, use the `node[N].reset()` command. Also use the `node[N].reset()` command to reset an instrument on a subordinate node.

When no value is specified for `system`, the default value is `true`.

You can only reset the entire system using `reset(true)` if the node is the master. If the node is not the master node, executing this command generates an error event.

Example

reset(true)	If the node is the master node, the entire system is reset; if the node is not the master node, an error event is generated.

Also see

[Resets](on page 2-153)

script.delete()

This function deletes a script from the run-time memory and nonvolatile memory.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
script.delete(scriptName)
```

scriptName	A string that represents the name of the script

Details

When a script is deleted, the global variable referring to this script is also deleted.

You must delete an existing script before you can use the name of that script again. Scripts are not automatically overwritten.

Example

`script.delete("test8")`	Deletes a user script named `test8` from nonvolatile memory and the global variable named `test8`.

Also see

script.load()

This function creates a script from a specified file.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
script.load(file)
scriptVar = script.load(file)
```

file	The path and file name of the script file to load; if *scriptVar* is not defined, this name is used as the global variable name for this script
scriptVar	The created script; a global variable with this name is used to reference the script

Details

The named that is used for *scriptVar* must not already exist as a global variable. In addition, the *scriptVar* name must be a global reference and not a local variable, table, or array.

For external scripts, the root folder of the USB flash drive has the absolute path `/usb1/`.

Example

`test8 = script.load("/usb1/testSetup.tsp")`	Loads the script with the file name `testSetup.tsp` that is on the USB flash drive and names it `test8`.

Also see

None

scriptVar.run()

This function runs a script.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
scriptVar.run()
scriptVar()
```

`scriptVar`	The name of the variable that references the script

Details

The `scriptVar.run()` function runs the script referenced by `scriptVar`. You can also run the script by using `scriptVar()`.

Example

`test8.run()`	Runs the script referenced by the variable `test8`.

Also see

None

scriptVar.save()

This function saves the script to nonvolatile memory or to a USB flash drive.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
scriptVar.save()
scriptVar.save(filename)
```

`scriptVar`	The name of variable that references the script
`filename`	The file name to use when saving the script to a USB flash drive

Details

The `scriptVar.save()` function saves a script to nonvolatile memory or a USB flash drive. The root folder of the USB flash drive has the absolute path `/usb1/`.

If no `filename` is specified, the script is saved to internal nonvolatile memory. If a `filename` is given, the script is saved to the USB flash drive.

If you set `scriptVar` to `autoexec`, the script is run when the instrument powers up. You must delete the existing `autoexec` script before saving the new one.

If no `filename` is specified (the filename parameter is an empty string), the script is saved to internal nonvolatile memory. If a `filename` is given, the script is saved to the USB flash drive.

You can add the file extension, but it is not required. The only allowed extension is `.tsp` (see Example 2).

Example 1

`test8.save()`	Saves the script referenced by the variable `test8` to nonvolatile memory.

Example 2

`test8.save("/usb1/myScript.tsp")`	Saves the script referenced by the variable `test8` to a file named `myScript.tsp` on your USB flash drive.

Also see

None

scriptVar.source

This attribute contains the source code of a script.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	No	Not applicable	Not applicable	Not applicable

Usage

```
code = scriptVar.source
```

code	The body of the script
scriptVar	The name of the variable that references the script that contains the source code

Details

The body of the script is a single string with lines separated by the new line character.

Example

```
print(test7.source)
```

Assuming a script named `test7` was created on the instrument, this example retrieves the source code.
Output:
```
reset()
display.settext(display.TEXT1, "Text on line 1")
display.settext(display.TEXT2, "Text on line 2")
```

Also see

scriptVar.save() (on page 8-236)

status.clear()

This function clears event registers and the event log.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status.clear()
```

Details

This command clears the event registers of the Questionable Event and Operation Event Register set. It does not affect the Questionable Event Enable or Operation Event Enable registers.

Example

status.clear()	Clear the bits in the registers

Also see

*CLS (on page B-2)

status.condition

This attribute stores the status byte condition register.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
statusByte = status.condition
```

statusByte	The status byte

Details

You can use this command to read the status byte, which is returned as a numeric value.

When an enabled status event occurs, a summary bit is set in this register to indicate the event occurrence. The returned value can indicate that one or more status events occurred. If more than one bit of the register is set, `statusByte` equals the sum of their decimal weights. For example, if 129 is returned, bits B0 and B7 are set (1 + 128). See Understanding bit settings (on page C-14) for additional information about reading bit values.

> NOTE
>
> If you are using the GPIB, USB, or VXI-11 serial poll sequence of the Model DMM7510 to get the status byte (also called a serial poll byte), B6 is the Request for Service (RQS) bit. If the bit is set, it indicates that a serial poll (SRQ) has occurred. For additional detail, see Serial polling and SRQ (on page C-12)

The meanings of the individual bits of this register are shown in the following table.

Bit	Decimal value	Constant	When set, indicates the following has occurred:
0	1	status.MSB	An enabled measurement event
1	2	Not used	
2	4	status.EAV	An error or status message is present in the Error Queue
3	8	status.QSB	An enabled questionable event
4	16	status.MAV	A response message is present in the Output Queue
5	32	status.ESB	An enabled standard event
6	64	status.MSS	An enabled summary bit of the status byte register is set
7	128	status.OSB	An enabled operation event

Example

`statusByte = status.condition` `print(statusByte)`	Returns `statusByte`. Example output: `1.29000e+02` Converting this output (129) to its binary equivalent yields 1000 0001 Therefore, this output indicates that the set bits of the status byte condition register are presently B0 (MSS) and B7 (OSB).

Also see

None

status.operation.condition

This attribute reads the Operation Event Register of the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
operationRegister = status.operation.condition
```

operationRegister	The status of the operation status register; a zero (0) indicates no bits set (also send 0 to clear all bits); other values indicate various bit settings

Details

This command reads the contents of the Operation Condition Register, which is one of the Operation Event Registers.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-14).

Example

`print(status.operation.condition)`	Returns the contents of the register.

Also see

Operation Event Register (on page C-7)

status.operation.enable

This attribute sets or reads the contents of the Operation Event Enable Register of the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	status.preset()	Nonvolatile memory	0

Usage

```
operationRegister = status.operation.enable
status.operation.enable = operationRegister
```

operationRegister	The status of the operation status register

Details

This command sets or reads the contents of the Enable register of the Operation Event Register.

When one of these bits is set, when the corresponding bit in the Operation Event Register or Operation Condition Register is set, the OSB bit in the Status Byte Register is set.

Example

`-- decimal 20480 = binary 0101 0000 0000 0000`
 	
`status.operation.enable = operationRegister`
	Sets the 12 and 14 bits of the operation status enable register using a decimal value.

Also see

Operation Event Register (on page C-7)
Understanding bit settings (on page C-14)

status.operation.event

This attribute reads the Operation Event Register of the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
operationRegister = status.operation.event
```

operationRegister	The status of the operation status register

Details

This attribute reads the operation event register of the status model.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Example

```status.operation.setmap(0, 4917, 4916)```
```defbuffer1.capacity = 10```
```dmm.measure.count = 10```
```dmm.measure.read()```
```print(status.operation.event)```
	Maps event number 4917 (Buffer Full) to set bit 0 in the Operation Event Register and event number 4916 (Buffer Empty) to clear bit 0. Resizes ```defbuffer1``` to 10 readings.
	Sets the measure count to 10 readings and makes a measurement.
	Reads the operation event register.
	Output:
	0

Also see

Operation Event Register (on page C-7)

status.operation.getmap()

This function requests the mapped set event and mapped clear event status for a bit in the Operation Event Registers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
setEvent, clearEvent = status.operation.getmap(bitNumber)
```

setEvent	The event mapped to set this bit; 0 if no mapping
clearEvent	The event mapped to clear this bit; 0 if no mapping
bitNumber	The bit number to check

Details

When you query the mapping for a specific bit, the instrument returns the events that were mapped to set and clear that bit. Zero (0) indicates that the bits have not been set.

Example

`print(status.operation.getmap(0))`	Query bit 0 of the Operation Event Register. Example output: `4917    4916`

Also see

Operation Event Register (on page C-7)
status.operation.setmap() (on page 8-242)

status.operation.setmap()

This function maps events to bits in the Operation Event Register.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status.operation.setmap(bitNumber, setEvent)
status.operation.setmap(bitNumber, setEvent, clearEvent)
```

bitNumber	The bit number that is mapped to an event (0 to 14)
setEvent	The number of the event that sets the bits in the condition and event registers; 0 if no mapping
clearEvent	The number of the event that clears the bit in the condition register; 0 if no mapping

Details

You can map events to bits in the event registers with this command. This allows you to cause bits in the condition and event registers to be set or cleared when the specified events occur. You can use any valid event number as the event that sets or clears bits.

When a mapped event is programmed to set bits, the corresponding bits in both the condition register and event register are set when the event is detected.

When a mapped event is programmed to clear bits, the bit in the condition register is set to 0 when the event is detected.

If the event is set to zero (0), the bit is never set.

Example

`status.operation.setmap(0, 2731, 2732)`	When event 2731(trigger model initiated) occurs, bit 0 in the condition and event registers of the Operation Event Register are set. When event 2732 (trigger model idled) occurs, bit 0 in the condition register is cleared.

Also see

Operation Event Register (on page C-7)
Programmable status register sets (on page C-5)
status.operation.getmap() (on page 8-241)

status.preset()

This function resets all bits in the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status.preset()
```

Details

This function clears the event registers and the enable registers for operation and questionable. It will not clear the Service Request Enable Register (`*SRE`) to Standard Request Enable Register (`*ESE`).

Preset does not affect the event queue.

The Standard Event Status Register is not affected by this command.

Example

`status.preset()`	Resets the instrument status model.

Also see

[Status model](#) (on page C-1)

status.questionable.condition

This attribute reads the Questionable Condition Register of the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
questionableRegister = status.questionable.condition
```

questionableRegister	The value of the register (0 to 65535)

Details

This command reads the contents of the Questionable Condition Register, which is one of the Questionable Event Registers.

For detail on interpreting the value of a register, see [Understanding bit settings](#) (on page C-14).

Example

`print(status.questionable.condition)`	Reads the Questionable Condition Register.

Also see

[Questionable Event Register](#) (on page C-7)
[Understanding bit settings](#) (on page C-14)

status.questionable.enable

This attribute sets or reads the contents of the questionable event enable register of the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	status.preset()	Nonvolatile memory	0

Usage

```
questionableRegister = status.questionable.enable
status.questionable.enable = questionableRegister
```

questionableRegister	The value of the register (0 to 65535)

Details

This command sets or reads the contents of the Enable register of the Questionable Event Register. When one of these bits is set, when the corresponding bit in the Questionable Event Register or Questionable Condition Register is set, the MSB and QSM bits in the Status Byte Register are set.

For detail on interpreting the value of a register, see Understanding bit settings (on page C-14).

Example

status.questionable.enable = 17 print(status.questionable.enable)	Set bits 0 and 4 of the Questionable Event Enable Register. Returns 17, which indicates the register was set correctly.

Also see

Questionable Event Register (on page C-7)

status.questionable.event

This attribute reads the Questionable Event Register.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
questionableRegister = status.questionable.event
```

questionableRegister	The value of the questionable status register (0 to 65535)

Details

This query reads the contents of the questionable status event register. After sending this command and addressing the instrument to talk, a value is sent to the computer. This value indicates which bits in the appropriate register are set.

The Questionable Register can be set to the numeric equivalent of the bit to set. To set more than one bit of the register, set the Questionable Register to the sum of their decimal weights. For example, to set bits B12 and B13, set the Questionable Register to 12,288 (which is the sum of 4,096 + 8,192).

Example 1

`-- decimal 66 = binary 0100 0010`
`questionableRegister = 66`
`status.questionable.enable = questionableRegister`
	Uses a decimal value to set bits B1 and B6 of the status questionable enable register.

Example 2

`-- decimal 2560 = binary 00001010 0000 0000`
`questionableRegister = 2560`
`status.questionable.enable = questionableRegister`
	Uses a decimal value to set bits B9 and B11 of the status questionable enable register.

Also see

Questionable Event Register (on page C-7)

status.questionable.getmap()

This function requests the mapped set event and mapped clear event status for a bit in the Questionable Event Registers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`setEvent, clearEvent = status.questionable.getmap(bitNumber)`	
`setEvent`	The event mapped to set this bit; 0 if no mapping
`clearEvent`	The event mapped to clear this bit; 0 if no mapping
`bitNumber`	The bit number to check (0 to 14)

Details

When you query the mapping for a specific bit, the instrument returns the events that were mapped to set and clear that bit. Zero (0) indicates that the bits have not been set.

Example

`print(status.questionable.getmap(9))`	Returns the events that were mapped to set and clear bit 9.

Also see

Questionable Event Register (on page C-7)
status.questionable.setmap() (on page 8-246)

status.questionable.setmap()

This function maps events to bits in the questionable event registers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status.questionable.setmap(bitNumber, setEvent)
status.questionable.setmap(bitNumber, setEvent, clearEvent)
```

bitNumber	The bit number that is mapped to an event (0 to 14)
setEvent	The number of the event that sets the bits in the condition and event registers; 0 if no mapping
clearEvent	The number of the event that clears the bit in the condition register; 0 if no mapping

Details

You can map events to bits in the event registers with this command. This allows you to cause bits in the condition and event registers to be set or cleared when the specified events occur. You can use any valid event number as the event that sets or clears bits.

When a mapped event is programmed to set bits, the corresponding bits in both the condition register and event register are set when the event is detected.

When a mapped event is programmed to clear bits, the bit in the condition register is set to 0 when the event is detected.

If the event is set to zero (0), the bit is never set.

Also see

[status.questionable.getmap()](status.questionable.getmap) (on page 8-245)

status.request_enable

This attribute stores the settings of the Service Request (SRQ) Enable Register.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	status.preset()	Not applicable	0

Usage

```
SRQEnableRegister = status.request_enable
status.request_enable = SRQEnableRegister
```

SRQEnableRegister	The status of the service request (SRQ) enable register; a zero (0) indicates no bits set (also send 0 to clear all bits); other values indicate various bit settings (0 to 255)

Details

This command sets or clears the individual bits of the Status Request Enable Register.

The Status Request Enable Register is cleared when power is cycled or when a parameter value of 0 is sent with this command.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Bit	Decimal value	Constants	When set, indicates the following has occurred:
0	1	`status.MSB`	An enabled event in the Measurement Event Register has occurred.
1	2	Not used	Not used.
2	4	`status.EAV`	An error or status message is present in the Error Queue.
3	8	`status.QSB`	An enabled event in the Questionable Status Register has occurred.
4	16	`status.MAV`	A response message is present in the Output Queue.
5	32	`status.ESB`	An enabled event in the Standard Event Status Register has occurred.
6	64	Not used	Not used.
7	128	`status.OSB`	An enabled event in the Operation Status Register has occurred.

Example 1

```requestSRQEnableRegister = status.MSB + status.OSB status.request_enable = requestSRQEnableRegister```	Uses constants to set the MSB and OSB bits of the service request (SRQ) enable register and clear all other bits.

Example 2

```-- decimal 129 = binary 10000001 requestSRQEnableRegister = 129 status.request_enable = requestSRQEnableRegister```	Uses a decimal value to set the MSB and OSB bits and clear all other bits of the service request (SRQ) enable register.

Example 3

```status.request_enable = 0```	Clear the register.

Also see

Status model (on page C-1)
Understanding bit settings (on page C-14)

status.standard.enable

This attribute reads or sets the bits in the Status Enable register of the Standard Event Register.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	status.preset()	Not applicable	0

Usage

```
standardRegister = status.standard.enable
status.standard.enable = standardRegister
```

standardRegister	The value of the Status Enable register of the Standard Event Register (0 to 255)

Details

When a bit in the Status Enable register is set on and the corresponding bit in the Standard Event Status register is set on, the ESB bit of the Status Byte Register is set to on.

To set a bit on, send the constant or value of the bit as the *standardRegister* parameter.

You can set the bit as a constant or a numeric value, as shown in the table below. To set more than one bit of the register, you can send multiple constants with + between them. You can also set *standardRegister* to the sum of their decimal weights. For example, to set bits B0 and B2, set *standardRegister* to 5 (which is the sum of 1 + 4). You can also send:

```
status.standard.enable = status.standard.OPC + status.standard.QYE
```

When zero (0) is returned, no bits are set. You can also send 0 to clear all bits.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Bit	Decimal value	Constant	When set, indicates the following has occurred:
0	1	`status.standard.OPC`	All pending selected instrument operations are complete and the instrument is ready to accept new commands. The bit is set in response to an [*OPC](#) (on page B-6) command or TSP [opc()](#) (on page 8-228) function.
1	2	Not used	Not used.
2	4	`status.standard.QYE`	Attempt to read data from an empty Output Queue.
3	8	Not used	Not used.
4	16	Not used	Not used.
5	32	Not used	Not used.
6	64	Not used	Not used.
7	128	`status.standard.PON`	The instrument has been turned off and turned back on since the last time this register was read.

Command errors include:

- **IEEE Std 488.2 syntax error:** The instrument received a message that does not follow the defined syntax of the IEEE Std 488.2 standard.
- **Semantic error:** The instrument received a command that was misspelled or received an optional IEEE Std 488.2 command that is not implemented in the instrument.
- **GET error:** The instrument received a Group Execute Trigger (GET) inside a program message.

Example 1

```
standardRegister = status.standard.OPC + status.standard.QYE
status.standard.enable = standardRegister
```
Uses constants to set the OPC and QYE bits of the standard event status enable register.

Example 2

```-- decimal 5 = binary 0000 0101```
```standardRegister = 5```
```status.standard.enable = standardRegister```
	Uses a decimal value to set the OPC and QYE bits of the standard event status enable register.

Also see

Standard Event Register (on page C-3)
Understanding bit settings (on page C-14)

status.standard.event

This attribute returns the contents of the Standard Event Status Register set of the status model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	status.preset()	Not applicable	0

Usage

```
standardRegister = status.standard.event
```

standardRegister	The status of the standard event status register

Details

When this command returns zero (0), no bits are set. You can send 0 to clear all bits.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Bit	Decimal value	Constant	When set, indicates the following has occurred:
0	1	`status.standard.OPC`	All pending selected instrument operations are complete and the instrument is ready to accept new commands. The bit is set in response to an *OPC (on page B-6) command or TSP opc() (on page 8-228) function.
1	2	Not used	Not used.
2	4	`status.standard.QYE`	Attempt to read data from an empty Output Queue.
3	8	Not used	Not used.
4	16	Not used	Not used.
5	32	Not used	Not used.
6	64	Not used	Not used.
7	128	`status.standard.PON`	The instrument has been turned off and turned back on since the last time this register was read.

Command errors include:

- **IEEE Std 488.2 syntax error:** The instrument received a message that does not follow the defined syntax of the IEEE Std 488.2 standard.
- **Semantic error:** The instrument received a command that was misspelled or received an optional IEEE Std 488.2 command that is not implemented in the instrument.
- **GET error:** The instrument received a Group Execute Trigger (GET) inside a program message.

Example

`print(status.standard.event)`	May return the value `129`, showing that the Standard Event Status Register contains binary `10000001`

Also see

Standard Event Register (on page C-3)
Understanding bit settings (on page C-14)

timer.cleartime()

This function resets the timer to zero (0) seconds.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
timer.cleartime()
```

Example

```dataqueue.clear()```
```dataqueue.add(35)```
```timer.cleartime()```
```delay(0.5)```
```dt = timer.gettime()```
```print("Delay time was " .. dt)```
```print(dataqueue.next())```
	Clear the data queue, add 35 to it, and then delay 0.5 seconds before reading it.
	Output:
	```Delay time was 0.500099```
	```35```

Also see

timer.gettime()

This function measures the elapsed time since the timer was last cleared.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
time = timer.gettime()
```

time	The elapsed time in seconds (1 μs resolution)

Example

```dataqueue.clear()```
```dataqueue.add(35)```
```timer.cleartime()```
```delay(0.5)```
```dt = timer.gettime()```
```print("Delay time was " .. dt)```
```print(dataqueue.next())```
	Clear the data queue, add 35 to it, and then delay 0.5 seconds before reading it.
	Output:
	```Delay time was 0.500099```
	```35```

Also see

trigger.blender[N].clear()

This function clears the blender event detector and resets the overrun indicator of blender *N*.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.blender[N].clear()
```

N	The blender number (1 or 2)

Details

This command sets the blender event detector to the undetected state and resets the overrun indicator of the event detector.

Example

`trigger.blender[2].clear()`	Clears the event detector for blender 2.

Also see

None

trigger.blender[N].orenable

This attribute selects whether the blender performs OR operations or AND operations.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger blender N reset	Configuration script	false (AND)

Usage

```
orenable = trigger.blender[N].orenable
trigger.blender[N].orenable = orenable
```

orenable	The type of operation: • `true`: OR operation • `false`: AND operation
N	The blender number (1 or 2)

Details

This command selects whether the blender waits for any one event (`OR`) or waits for all selected events (`AND`) before signaling an output event.

Example

`trigger.blender[1].orenable = true`
`trigger.blender[1].stimulus[1] = trigger.EVENT_DIGIO3`
`trigger.blender[1].stimulus[2] = trigger.EVENT_DIGIO5`
	Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.

Also see

trigger.blender[N].overrun

This attribute indicates whether or not an event was ignored because of the event detector state.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Instrument reset Trigger blender *N* clear Trigger blender *N* reset	Not applicable	Not applicable

Usage

`overrun = trigger.blender[N].overrun`	

`overrun`	Trigger blender overrun state (`true` or `false`)
`N`	The blender number (1 or 2)

Details

Indicates if an event was ignored because the event detector was already in the detected state when the event occurred. This is an indication of the state of the event detector that is built into the event blender itself.

This command does not indicate if an overrun occurred in any other part of the trigger model or in any other trigger object that is monitoring the event. It also is not an indication of an action overrun.

Example

`print(trigger.blender[1].overrun)`	If an event was ignored, the output is `true`. If an event was not ignored, the output is `false`.

Also see

trigger.blender[N].reset()

This function resets some of the trigger blender settings to their factory defaults.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`trigger.blender[N].reset()`	

`N`	The trigger event blender (1 or 2)

Details

The `trigger.blender[N].reset()` function resets the following attributes to their factory defaults:

- `trigger.blender[N].orenable`
- `trigger.blender[N].stimulus[M]`

It also clears `trigger.blender[N].overrun`.

Example

`trigger.blender[1].reset()`	Resets the trigger blender 1 settings to factory defaults.

Also see

trigger.blender[N].stimulus[M]

This attribute specifies the events that trigger the blender.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger blender N reset	Configuration script	trigger.EVENT_NONE

Usage

```
event = trigger.blender[N].stimulus[M]
trigger.blender[N].stimulus[M] = event
```

event	The event that triggers the blender action; see **Details**
N	An integer that represents the trigger event blender (1 or 2)
M	An integer representing the stimulus index (1 to 4)

Details

There are four stimulus inputs that can each select a different event.
Use zero to disable the blender input.

The *event* parameter may be any of the trigger events shown in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block *N* (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line *N* (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line *N* (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object *N* (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender *N* (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer *N* (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

`digio.line[3].mode = digio.MODE_TRIGGER_IN`
`digio.line[5].mode = digio.MODE_TRIGGER_IN`
`trigger.digin[3].edge = trigger.EDGE_FALLING`
`trigger.digin[5].edge = trigger.EDGE_FALLING`
`trigger.blender[1].orenable = true`
`trigger.blender[1].stimulus[1] = trigger.EVENT_DIGIO3`
`trigger.blender[1].stimulus[2] = trigger.EVENT_DIGIO5`
	Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.

Also see

trigger.blender[N].reset() (on page 8-253)

trigger.blender[N].wait()

This function waits for a blender trigger event to occur.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`triggered = trigger.blender[N].wait(timeout)`

triggered	Trigger detection indication for blender
N	The trigger blender (1 or 2) on which to wait
timeout	Maximum amount of time in seconds to wait for the trigger blender event

Details

This function waits for an event blender trigger event. If one or more trigger events were detected since the last time `trigger.blender[`*`N`*`].wait()` or `trigger.blender[`*`N`*`].clear()` was called, this function returns immediately.

After detecting a trigger with this function, the event detector automatically resets and rearms. This is true regardless of the number of events detected.

Example

| ```
digio.line[3].mode = digio.MODE_TRIGGER_IN
digio.line[5].mode = digio.MODE_TRIGGER_IN
trigger.digin[3].edge = trigger.EDGE_FALLING
trigger.digin[5].edge = trigger.EDGE_FALLING
trigger.blender[1].orenable = true
trigger.blender[1].stimulus[1] = trigger.EVENT_DIGIO3
trigger.blender[1].stimulus[2] = trigger.EVENT_DIGIO5
print(trigger.blender[1].wait(3))
``` | Generate a trigger blender 1 event when a digital I/O trigger happens on line 3 or 5.<br>Wait 3 s while checking if trigger blender 1 event has occurred. |

Also see

trigger.clear()

This function clears any pending command triggers.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.clear()
```

Details

A command trigger indicates if a trigger event has been detected over a command interface since the last `trigger.wait()` command was sent.

`trigger.clear()` clears the command triggers and discards the history of trigger events.

Also see

trigger.digin[N].clear()

This function clears the trigger event on a digital input line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.digin[N].clear()
```

N	Digital I/O trigger line (1 to 6)

Details

The event detector of a trigger enters the detected state when an event is detected. For the specified trigger line, this command clears the event detector, discards the history, and clears the overrun status(sets the overrun status to `false`).

Example

`trigger.digin[2].clear()`	Clears the trigger event detector on I/O line 2.

Also see

digio.line[N].mode (on page 8-48)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].overrun (on page 8-258)
trigger.digin[N].wait() (on page 8-259)

trigger.digin[N].edge

This attribute sets the edge used by the trigger event detector on the given trigger line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	trigger.EDGE_FALLING

Usage

```
detectedEdge = trigger.digin[N].edge
trigger.digin[N].edge = detectedEdge
```

detectedEdge	The trigger logic value: • Detect falling-edge triggers as inputs: `trigger.EDGE_FALLING` • Detect rising-edge triggers as inputs: `trigger.EDGE_RISING` • Detect either falling or rising-edge triggers as inputs: `trigger.EDGE_EITHER` • See **Details** for descriptions of values
N	Digital I/O trigger line (1 to 6)

Details

The input state of the digital I/O line is controlled by the type of edge specified by this command.

To directly control the line state, set the mode of the line to digital and use the write command. When the digital line mode is set for open drain, the edge settings assert a TTL low-pulse.

Trigger mode values

Value	Description
`trigger.EDGE_FALLING`	Detects falling-edge triggers as input when the line is configured as an input or open drain
`trigger.EDGE_RISING`	Detects rising-edge triggers as input when the line is configured as an open drain
`trigger.EDGE_EITHER`	Detects rising- or falling-edge triggers as input when the line is configured as an input or open drain

Example

`digio.line[4].mode = digio.MODE_TRIGGER_IN` `trigger.digin[4].edge = trigger.EDGE_RISING`	Sets the trigger mode for digital I/O line 4 to detect a rising-edge trigger as an input.

Also see

digio.line[N].mode (on page 8-48)
digio.line[N].reset() (on page 8-50)
digio.writeport() (on page 8-52)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].clear() (on page 8-257)

trigger.digin[N].overrun

This attribute returns the event detector overrun status.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Digital I/O trigger N clear Digital I/O trigger N reset	Not applicable	Not applicable

Usage

```
overrun = trigger.digin[N].overrun
```

overrun	Trigger overrun state (`true` or `false`)
N	Digital I/O trigger line (1 to 6)

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 = trigger.digin[1].overrun`
`print(overrun)` | If there is no trigger overrun on digital input 1, the output is:
`false` |

Also see

digio.line[N].mode (on page 8-48)
digio.line[N].reset() (on page 8-50)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].clear() (on page 8-257)

trigger.digin[N].wait()

This function waits for a trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`triggered = trigger.digin[N].wait(timeout)`

`triggered`	Trigger detected: `true` No triggers detected during the timeout period: `false`
`N`	Digital I/O trigger line (1 to 6)
`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 `trigger.digin[N].wait()` or `trigger.digin[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

`digio.line[4].mode = digio.MODE_TRIGGER_IN`
`triggered = trigger.digin[4].wait(3)`
`print(triggered)`
	Waits up to 3 s for a trigger to be detected on trigger line 4, then outputs the results.
	Output if no trigger is detected:
	`false`
	Output if a trigger is detected:
	`true`

Also see

digio.line[N].mode (on page 8-48)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].clear() (on page 8-257)

trigger.digout[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

`trigger.digout[N].assert()`	
N	Digital I/O trigger line (1 to 6)

Details

Initiates a trigger event and does not wait for completion. The pulse width that is set determines how long the instrument asserts the trigger.

Example

`digio.line[2].mode = digio.MODE_TRIGGER_OUT`
`trigger.digout[2].pulsewidth = 20e-6`
`trigger.digout[2].assert()`
	Asserts a trigger on digital I/O line 2 with a pulse width of 20 µs.

Also see

digio.line[N].mode (on page 8-48)
Digital I/O port configuration (on page 3-45)
trigger.digout[N].pulsewidth (on page 8-261)
trigger.digout[N].pulsewidth (on page 8-261)

trigger.digout[N].logic

This attribute sets the output logic of the trigger event generator to positive or negative for the specified 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	trigger.LOGIC_NEGATIVE

Usage

```
logicType = trigger.digout[N].logic
trigger.digout[N].logic = logicType
```

logicType	The output logic of the trigger generator: • Assert a TTL-high pulse for output: `trigger.LOGIC_POSITIVE` • Assert a TTL-low pulse for output: `trigger.LOGIC_NEGATIVE`
N	Digital I/O trigger line (1 to 6)

Details

This attribute controls the logic that the output trigger generator uses on the given trigger line.

The output state of the digital I/O line is controlled by the trigger logic, and the user-specified output state of the line is ignored.

Example

`digio.line[4].mode = digio.MODE_TRIGGER_OUT` `trigger.digout[4].logic = trigger.LOGIC_NEGATIVE`	Sets line 4 mode to be a trigger output and sets the output logic of the trigger event generator to negative (asserts a low pulse).

Also see

digio.line[N].mode (on page 8-48)
digio.line[N].reset() (on page 8-50)
Digital I/O port configuration (on page 3-45)

trigger.digout[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	Restore configuration Instrument reset Power cycle Digital I/O trigger N reset	Configuration script	10e-6 (10 μs)

Usage

```
width = trigger.digout[N].pulsewidth
trigger.digout[N].pulsewidth = width
```

width	The pulse width (0 to 100,000 s)
N	Digital I/O trigger line (1 to 6)

Details

Setting the pulse width to zero (0) seconds asserts the trigger indefinitely. To release the trigger line, use `trigger.digout[N].release()`.

Example

`digio.line[4].mode = digio.MODE_TRIGGER_OUT` `trigger.digout[4].pulsewidth = 20e-6`	Sets the pulse width for trigger line 4 to 20 µs.

Also see

digio.line[N].mode (on page 8-48)
digio.line[N].reset() (on page 8-50)
Digital I/O port configuration (on page 3-45)
trigger.digout[N].assert() (on page 8-260)
trigger.digout[N].release() (on page 8-262)

trigger.digout[N].release()

This function releases an indefinite length or latched trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`trigger.digout[N].release()`

N	Digital I/O trigger line (1 to 6)

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.

Example

`digio.line[4].mode = digio.MODE_TRIGGER_OUT` `trigger.digout[4].release()`	Releases digital I/O trigger line 4.

Also see

digio.line[N].mode (on page 8-48)
Digital I/O port configuration (on page 3-45)
trigger.digout[N].assert() (on page 8-260)
trigger.digout[N].pulsewidth (on page 8-261)

trigger.digout[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	trigger.EVENT_NONE

Usage

```
event = trigger.digout[N].stimulus
trigger.digout[N].stimulus = event
```

event	The event to use as a stimulus; see **Details**
N	Digital I/O trigger line (1 to 6)

Details

The digital trigger pulsewidth command determines how long the trigger is asserted.

The trigger stimulus for a digital I/O line can be set to one of the trigger events that are described in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	trigger.EVENT_NONE
Front-panel TRIGGER key press	trigger.EVENT_DISPLAY
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	trigger.EVENT_NOTIFYN
A command interface trigger: • Any remote interface: *TRG • GPIB only: GET bus command • VXI-11: VXI-11 command device_trigger	trigger.EVENT_COMMAND
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	trigger.EVENT_DIGION
Line edge detected on TSP-Link synchronization line N (1 to 3)	trigger.EVENT_TSPLINKN
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	trigger.EVENT_LANN
Trigger event blender N (1 to 2), which combines trigger events	trigger.EVENT_BLENDERN
Trigger timer N (1 to 4) expired	trigger.EVENT_TIMERN
Analog trigger	trigger.EVENT_ANALOGTRIGGER
External in trigger	trigger.EVENT_EXTERNAL

Example

`digio.line[2].mode = digio.MODE_TRIGGER_OUT` `trigger.digout[2].stimulus = trigger.EVENT_TIMER3`	Set the stimulus for output digital trigger line 2 to be the expiration of trigger timer 3.

Also see

digio.line[N].mode (on page 8-48)
digio.line[N].reset() (on page 8-50)
Digital I/O port configuration (on page 3-45)
trigger.digin[N].clear() (on page 8-257)
trigger.digout[N].assert() (on page 8-260)

trigger.ext.reset()

This function resets the edge, logic, and stimulus values for the external in/out port to their default values.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.ext.reset()
```

Details

This function resets the following attributes to their default values:

- `trigger.extin.edge`

- `trigger.extout.logic`

- `trigger.extout.stimulus`

It also clears `trigger.extin.overrun`.

Example

```
-- Set the external I/O trigger line for a falling edge
trigger.extin.edge = trigger.EDGE_RISING
-- Set the logic to negative
trigger.extout.logic = trigger.LOGIC_NEGATIVE
--Set the stimulus to timer 3
trigger.extout.stimulus = trigger.EVENT_TIMER3
-- Print configuration (before reset)
print(trigger.extin.edge, trigger.extout.logic, trigger.extout.stimulus)
-- Reset the external in/out line to default values.
trigger.ext.reset()
-- Print configuration (after reset)
print(trigger.extin.edge, trigger.extout.logic, trigger.extout.stimulus)
```
Output before reset:
`trigger.EDGE_RISING     trigger.LOGIC_NEGATIVE   trigger.EVENT_TIMER3`
Output after reset:
`trigger.EDGE_FALLING    trigger.LOGIC_NEGATIVE   trigger.EVENT_NONE`

Also see

trigger.extin.edge (on page 8-265)
trigger.extout.logic (on page 8-268)
trigger.extout.stimulus (on page 8-268)

trigger.extin.clear()

This function clears the trigger event on the external in line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.extin.clear()
```

Details

The event detector of a trigger enters the detected state when an event is detected. This command clears the event detector, discards the history, and clears the overrun status (sets the overrun status to false).

Example

`trigger.extin.clear()`	Clears the trigger event detector.

Also see

[trigger.extin.overrun](#) (on page 8-266)

trigger.extin.edge

This attribute sets the type of edge that is detected as an input on the external in line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	trigger.EDGE_FALLING

Usage

```
detectedEdge = trigger.extin.edge
trigger.extin.edge = detectedEdge
```

detectedEdge	The trigger edge value: • Detect falling-edge triggers as inputs: `trigger.EDGE_FALLING` • Detect rising-edge triggers as inputs: `trigger.EDGE_RISING` • Detect either falling or rising-edge triggers as inputs: `trigger.EDGE_EITHER` See **Details** for descriptions of values

Details

The input state of the external I/O line is controlled by the type of edge specified by this command.

Trigger mode values

Value	Description
`trigger.EDGE_FALLING`	Detects falling-edge triggers as input
`trigger.EDGE_RISING`	Detects rising-edge triggers as input
`trigger.EDGE_EITHER`	Detects rising- or falling-edge triggers as input

Example

`trigger.extin.edge = trigger.EDGE_RISING`	Sets the external I/O to detect a rising-edge trigger as an input.

Also see

trigger.extin.overrun

This attribute returns the event detector overrun status.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Instrument reset External I/O reset	Not saved	Not applicable

Usage

```
overrun = trigger.extin.overrun
```

`overrun`	Trigger overrun state (`true` or `false`)

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 = trigger.extin.overrun` `print(overrun)`	If there is no trigger overrun on the external in, the output is: `false`

Also see

trigger.extin.wait()

This function waits for a trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
triggered = trigger.extin.wait(timeout)
```

`triggered`	Trigger detected: `true` No triggers detected during the timeout period: `false`
`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 `trigger.extin.wait()` or `trigger.extin.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 = trigger.extin.wait(3)`
`print(triggered)`
	Waits up to 3 s for a trigger to be detected on the external trigger line, then outputs the results.
	Output if no trigger is detected:
	`false`
	Output if a trigger is detected:
	`true`

Also see

[trigger.extin.clear()](#) (on page 8-265)

trigger.extout.assert()

This function asserts a trigger on the external I/O line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`trigger.extout.assert()`

Details

Initiates a trigger event and does not wait for completion.

Example

`trigger.extout.assert()`	Asserts a trigger on the external out line2.

Also see

None

trigger.extout.logic

This attribute sets the output logic of the trigger event generator to positive or negative for the external out 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	trigger.LOGIC_POSITIVE

Usage

```
logicType = trigger.extout.logic
trigger.extout[N].logic = logicType
```

logicType	The output logic of the trigger generator: • Assert a TTL-high pulse for output: `trigger.LOGIC_POSITIVE` • Assert a TTL-low pulse for output: `trigger.LOGIC_NEGATIVE`

Details

This command sets the trigger event generator to assert a TTL pulse for output logic. Positive is a high pulse; negative is a low pulse.

Example

```
trigger.ext.reset()
trigger.extin.clear()
trigger.extout.logic = trigger.LOGIC_NEGATIVE
trigger.extout.stimulus = trigger.EVENT_EXTERNAL
trigger.extin.edge = trigger.EDGE_FALLING
```

Reset the external I/O port values to their defaults. Clear any event triggers on the external in line. Set the output logic to negative (it asserts a low pulse). Set the stimulus to the external input. Set the external input to detect a falling edge.

Also see

trigger.ext.reset() (on page 8-264)

trigger.extout.stimulus

This attribute selects the event that causes a trigger to be asserted on the external 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	trigger.EVENT_NONE

Usage

```
event = trigger.extout.stimulus
trigger.extout.stimulus = event
```

event	The event to use as a stimulus; see **Details**

Details

The trigger stimulus for the external output line can be set to one of the trigger events described in the following table.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender N (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer N (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

`trigger.extout.stimulus = trigger.EVENT_TIMER3`	Set the stimulus for the external output to be the expiration of trigger timer 3.

Also see

[trigger.extin.edge](#) (on page 8-265)
[trigger.extin.wait()](#) (on page 8-266)
[trigger.extout.assert()](#) (on page 8-267)
[trigger.extout.logic](#) (on page 8-268)

trigger.lanin[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

```
trigger.lanin[N].clear()
```

N	The LAN event number (1 to 8) to clear

Details

The trigger event detector enters the detected state when an event is detected. This function clears a trigger event detector and discards the previous of the trigger packet.
This function clears all overruns associated with this LAN trigger.

Example

`trigger.lanin[5].clear()`	Clears the event detector with LAN event trigger 5.

Also see

[trigger.lanin[N].overrun](#) (on page 8-271)

trigger.lanin[N].edge

This attribute sets the trigger operation and detection mode of the specified LAN event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	trigger.EDGE_EITHER

Usage

```
edgeMode = trigger.lanin[N].edge
trigger.lanin[N].edge = edgeMode
```

edgeMode	The trigger mode; see the **Details** for more information
N	The LAN event number (1 to 8)

Details

This command controls how the trigger event detector and the output trigger generator operate on the given trigger. These settings are intended to provide behavior similar to the digital I/O triggers.

LAN trigger mode values		
Mode	**Trigger packets detected as input**	**LAN trigger packet generated for output with a…**
`trigger.EDGE_EITHER`	Rising or falling edge (positive or negative state)	negative state
`trigger.EDGE_FALLING`	Falling edge (negative state)	negative state
`trigger.EDGE_RISING`	Rising edge (positive state)	positive state

Example

`trigger.lanin[1].edge = trigger.EDGE_FALLING`	Set the edge state of LAN event 1 to falling.

Also see

Digital I/O (on page 3-43)
TSP-Link system expansion interface (on page 3-102)

trigger.lanin[N].overrun

This attribute contains the overrun status of the LAN event detector.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	LAN trigger *N* clear	Not applicable	Not applicable

Usage

`overrun = trigger.lanin[N].overrun`

overrun	The trigger overrun state for the specified LAN packet (`true` or `false`)
N	The LAN event number (1 to 8)

Details

This command indicates whether an event has been ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the synchronization line itself. It does not indicate if an overrun occurred in any other part of the trigger model, or in any other construct that is monitoring the event.

It also is not an indication of an output trigger overrun.

Example

`overrun = trigger.lanin[5].overrun` `print(overrun)`	Checks the overrun status of a trigger on LAN5 and outputs the value, such as: `false`

Also see

trigger.lanin[N].clear() (on page 8-270)
trigger.lanin[N].wait() (on page 8-272)
trigger.lanout[N].assert() (on page 8-272)
trigger.lanout[N].stimulus (on page 8-278)

trigger.lanin[N].wait()

This function waits for an input trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
triggered = trigger.lanin[N].wait(timeout)
```

triggered	Trigger detection indication (true or false)
N	The trigger packet over LAN to wait for (1 to 8)
timeout	Maximum amount of time in seconds to wait for the trigger event

Details

If one or more trigger events have been detected since the last time `trigger.lanin[N].wait()` or `trigger.lanin[N].clear()` was called, this function returns immediately.

After waiting for a LAN trigger event with this function, the event detector is automatically reset and rearmed regardless of the number of events detected.

Example

`triggered = trigger.lanin[5].wait(3)`	Wait for a trigger event with LAN trigger 5 with a timeout of 3 s.

Also see

trigger.lanin[N].clear() (on page 8-270)
trigger.lanin[N].overrun (on page 8-271)
trigger.lanout[N].assert() (on page 8-272)
trigger.lanout[N].stimulus (on page 8-278)

trigger.lanout[N].assert()

This function simulates the occurrence of the trigger and generates the corresponding event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.lanout[N].assert()
```

N	The LAN event number (1 to 8)

Details

Generates and sends a LAN trigger packet for the LAN event number specified.

Sets the pseudo line state to the appropriate state.

The following indexes provide the listed LXI events:

- 1:LAN0
- 2:LAN1
- 3:LAN2
- …
- 8:LAN7

Example

`trigger.lanout[5].assert()`	Creates a trigger with LAN trigger 5.

Also see

lan.lxidomain (on page 8-216)
trigger.lanin[N].clear() (on page 8-270)
trigger.lanin[N].overrun (on page 8-271)
trigger.lanin[N].wait() (on page 8-272)
trigger.lanout[N].assert() (on page 8-272)
trigger.lanout[N].ipaddress (on page 8-276)
trigger.lanout[N].protocol (on page 8-277)
trigger.lanout[N].stimulus (on page 8-278)

trigger.lanout[N].connect()

This function prepares the event generator for outgoing trigger events.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.lanout[N].connect()
```

N	The LAN event number (1 to 8)

Details

This command prepares the event generator to send event messages. For TCP connections, this opens the TCP connection.

The event generator automatically disconnects when either the protocol or IP address for this event is changed.

Example

| ```
trigger.lanout[1].protocol =
    lan.PROTOCOL_MULTICAST
trigger.lanout[1].connect()
trigger.lanout[1].assert()
``` | Set the protocol for LAN trigger 1 to be multicast when sending LAN triggers. Then, after connecting the LAN trigger, send a message on LAN trigger 1 by asserting it. |
|---|---|

Also see

trigger.lanin[N].overrun (on page 8-271)
trigger.lanin[N].wait() (on page 8-272)
trigger.lanout[N].assert() (on page 8-272)
trigger.lanout[N].ipaddress (on page 8-276)
trigger.lanout[N].protocol (on page 8-277)
trigger.lanout[N].stimulus (on page 8-278)

trigger.lanout[N].connected

This attribute contains the LAN event connection state.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Not applicable	Not applicable	Not applicable

Usage

```
connected = trigger.lanout[N].connected
```

connected	The LAN event connection state: • `true`: Connected • `false`: Not connected
N	The LAN event number (1 to 8)

Details

This is set to `true` when the LAN trigger is connected and ready to send trigger events after a successful `trigger.lanout[N].connect()` command. If the LAN trigger is not ready to send trigger events, this value is `false`.

This attribute is also `false` when the `trigger.lanout[N].protocol` or `trigger.lanout[N].ipaddress` attribute is changed or when the remote connection closes the connection.

Example

`trigger.lanout[1].protocol = lan.PROTOCOL_MULTICAST` `print(trigger.lanout[1].connected)`	Outputs `true` if connected, or `false` if not connected. Example output: `false`

Also see

trigger.lanout[N].connect() (on page 8-274)
trigger.lanout[N].ipaddress (on page 8-276)
trigger.lanout[N].protocol (on page 8-277)

trigger.lanout[N].disconnect()

This function disconnects the LAN trigger event generator.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`trigger.lanout[N].disconnect()`	
N	The LAN event number (1 to 8)

Details

When this command is set for TCP connections, this closes the TCP connection.

The LAN trigger automatically disconnects when either the `trigger.lanout[N].protocol` or `trigger.lanout[N].ipaddress` attributes for this event are changed.

Also see

trigger.lanout[N].ipaddress (on page 8-276)
trigger.lanout[N].protocol (on page 8-277)

trigger.lanout[N].ipaddress

This attribute specifies the address (in dotted-decimal format) of UDP or TCP listeners.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	"0.0.0.0"

Usage

```
ipAddress = trigger.lanout[N].ipaddress
trigger.lanout[N].ipaddress = ipAddress
```

ipAddress	The LAN address for this attribute as a string in dotted decimal notation
N	The LAN event number (1 to 8)

Details

Sets the IP address for outgoing trigger events.

After you change this setting, you must send the connect command before outgoing messages can be sent.

Example

`trigger.lanout[3].protocol = lan.PROTOCOL_TCP`
`trigger.lanout[3].ipaddress = "192.0.32.10"`
`trigger.lanout[3].connect()`
	Set the protocol for LAN trigger 3 to be TCP when sending LAN triggers.
	Use IP address `"192.0.32.10"` to connect the LAN trigger.

Also see

trigger.lanout[N].connect() (on page 8-274)

trigger.lanout[N].logic

This attribute sets the logic on which the trigger event detector and the output trigger generator operate on the given trigger line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	trigger.LOGIC_NEGATIVE

Usage

```
logicType = trigger.lanout[N].logic
trigger.lanout[N].logic = logicType
```

logicType	The type of logic: • Positive: trigger.LOGIC_POSITIVE • Negative: trigger.LOGIC_NEGATIVE
N	The LAN event number (1 to 8)

Example

trigger.lanout[2].logic = trigger.LOGIC_POSITIVE	Set the logic for LAN trigger line 2 to positive.

Also see

None

trigger.lanout[N].protocol

This attribute sets the LAN protocol to use for sending trigger messages.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	lan.PROTOCOL_TCP

Usage

```
protocol = trigger.lanout[N].protocol
trigger.lanout[N].protocol = protocol
```

protocol	The protocol to use for messages from the trigger: • lan.PROTOCOL_TCP • lan.PROTOCOL_UDP • lan.PROTOCOL_MULTICAST
N	The LAN event number (1 to 8)

Details

The LAN trigger listens for trigger messages on all the supported protocols. However, it uses the designated protocol for sending outgoing messages.

After you change this setting, you must re-connect the LAN trigger event generator before you can send outgoing event messages.

When multicast is selected, the trigger IP address is ignored and event messages are sent to the multicast address 224.0.23.159.

Example

| print(trigger.lanout[1].protocol) | Get LAN protocol that is being used for sending trigger messages for LAN event 1. |

Also see

[trigger.lanout[N].connect()](#) (on page 8-274)
[trigger.lanout[N].ipaddress](#) (on page 8-276)

trigger.lanout[N].stimulus

This attribute specifies events that cause this trigger to assert.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle	Configuration script	trigger.EVENT_NONE

Usage

```
LANevent = trigger.lanout[N].stimulus
trigger.lanout[N].stimulus = LANevent
```

| LANevent | The LAN event that causes this trigger to assert; see **Details** for values |
| N | A number specifying the trigger packet over the LAN for which to set or query the trigger source (1 to 8) |

Details

This attribute specifies which event causes a LAN trigger packet to be sent for this trigger. Set the event to one of the existing trigger events, which are shown in the following table.

Setting this attribute to none disables automatic trigger generation.

If any events are detected before the trigger LAN connection is sent, the event is ignored and the action overrun is set.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender N (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer N (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

`trigger.lanout[5].stimulus = trigger.EVENT_TIMER1`	Use the timer 1 trigger event as the source for LAN trigger 5 stimulus.

Also see

trigger.lanout[N].connect() (on page 8-274)
trigger.lanout[N].ipaddress (on page 8-276)

trigger.model.abort()

This function stops all trigger model commands on the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.abort()
```

Details

When this command is received, the instrument stops the trigger model.

Example

`trigger.model.abort()`	Terminates all commands related to the trigger model on the instrument.

Also see

Effect of GPIB line events on Model DMM7510 (on page 2-66)
Aborting the trigger model (on page 3-95)
Trigger model (on page 3-72)

trigger.model.getblocklist()

This function returns the settings for all trigger model building blocks.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.getblocklist()
```

Details

This returns the settings for the trigger model.

Example

```
print(trigger.model.getblocklist())
```
Returns the settings for the trigger model. Example output is:
```
 1) BUFFER_CLEAR          BUFFER: defbuffer1
 2) MEASURE               BUFFER: defbuffer1 COUNT: 1
 3) BRANCH_COUNTER        VALUE: 5  BRANCH_BLOCK: 2
 4) DELAY_CONSTANT        DELAY: 1.000000000
 5) BRANCH_COUNTER        VALUE: 3  BRANCH_BLOCK: 2
```

Also see

trigger.model.getbranchcount() (on page 8-281)

trigger.model.getbranchcount()

This function returns the count value of the trigger model counter block.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.getbranchcount(blockNumber)
```

blockNumber	The sequence of the block in the trigger model

Details

This command returns the counter value. When the counter is active, this returns the present count. If the trigger model has started or is running but has not yet reached the counter block, this value is 0.

Example

`print(trigger.model.getbranchcount(4))`	Returns the value of the counter for building block 4.

Also see

[trigger.model.setblock() — trigger.BLOCK_BRANCH_COUNTER](#) (on page 8-297)

trigger.model.initiate()

This function starts the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.initiate()
```

Also see

[Trigger model](#) (on page 3-72)
[trigger.model.abort()](#) (on page 8-280)

trigger.model.load() — ConfigList

This function loads a predefined trigger model configuration that uses a measure configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("ConfigList", measureConfigList)
trigger.model.load("ConfigList", measureConfigList, delay)
trigger.model.load("ConfigList", measureConfigList, delay, bufferName)
trigger.model.load("ConfigList", measureConfigList, delay, bufferName,
    readingBlock)
```

measureConfigList	A string that contains the name of the measurement configuration list to use
delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; defaults to `defbuffer1`
readingBlock	Define a measure or digitize block for the trigger model; options are: • `trigger.READING_ACTIVE`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `trigger.READING_ACTIVE` is used • `trigger.READING_MEASURE`: Adds a measure block to the trigger model • `trigger.READING_DIGITIZE`: Adds a digitize block to the trigger model

Details

This trigger model template incorporates a configuration list.

You must set up the configuration lists before loading the trigger model.

You can also set a delay and change the reading buffer.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

```
reset()
dmm.measure.func = dmm.FUNC_AC_CURRENT
dmm.measure.configlist.create("MEASURE_LIST")
dmm.measure.range = 1e-3
dmm.measure.configlist.store("MEASURE_LIST")
dmm.measure.range = 10e-3
dmm.measure.configlist.store("MEASURE_LIST")
dmm.measure.range = 100e-3
dmm.measure.configlist.store("MEASURE_LIST")
trigger.model.load("ConfigList", "MEASURE_LIST")
trigger.model.initiate()
waitcomplete()
printbuffer(1, defbuffer1.n, defbuffer1.readings)
```

Reset the instrument.
Set the measure function to AC current.
Set up a configuration list named `MEASURE_LIST`.
Load the configuration list trigger model, using the points in this configuration list.
Start the trigger model.
Wait for the trigger model to complete.
Return the results from the reading buffer.
Example output:
`9.9246953126e-07, 6.9921188254e-06, 3.8904102673e-05`

Also see

[dmm.measure.configlist.create()](#) (on page 8-134)

trigger.model.load() — DurationLoop

This function loads a predefined trigger model configuration that makes continuous measurements for a specified amount of time.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("DurationLoop", duration)
trigger.model.load("DurationLoop", duration, delay)
trigger.model.load("DurationLoop", duration, delay, bufferName)
trigger.model.load("DurationLoop", duration, delay, bufferName, readingBlock)
```

duration	The amount of time for which to make measurements (500 ns to 100,000 s)
delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; defaults to `defbuffer1`
readingBlock	Define a measure or digitize block for the trigger model; options are: • `trigger.READING_ACTIVE`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `trigger.READING_ACTIVE` is used • `trigger.READING_MEASURE`: Adds a measure block to the trigger model • `trigger.READING_DIGITIZE`: Adds a digitize block to the trigger model

Details

When you load this predefined trigger model, you can specify amount of time to make a measurement and the length of the delay before the measurement.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

```
reset()
-- Set up measure function
dmm.measure.func = dmm.FUNC_DC_CURRENT
-- Initiate readings
trigger.model.load("DurationLoop", 10, 0.01)
trigger.model.initiate()
```

Reset the instrument.
Set the instrument to measure current.
Load the duration loop trigger model to take measurements for 10 s with a 10 ms delay before each measurement.
Start the trigger model.

Also see

None

trigger.model.load() — Empty

This function clears the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("Empty")
```

Details

When you load this predefined trigger model, any blocks that have been defined in the trigger model are cleared.

Example

```
trigger.model.load("Empty")
```

Reset trigger model settings.

Also see

None

trigger.model.load() — ExtDigTrigger

This function loads a predefined trigger model configuration that sets up an external trigger through the digital I/O or external I/O line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("ExtDigTrigger", digInLine, digOutLine, count)
trigger.model.load("ExtDigTrigger", digInLine, digOutLine, count, delay)
trigger.model.load("ExtDigTrigger", digInLine, digOutLine, count, delay,
    bufferName)
trigger.model.load("ExtDigTrigger", digInLine, digOutLine, count, delay,
    bufferName, readingBlock)
```

digInLine	The digital input line (1 to 6) or external input line (7); also the event on which the trigger model will wait in block 1
digOutLine	The digital output line (1 to 6) or external output line (7)
count	The number of measurements the instrument will make
delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; defaults to `defbuffer1`
readingBlock	Define a measure or digitize block for the trigger model; options are: `trigger.READING_ACTIVE`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `trigger.READING_ACTIVE` is used `trigger.READING_MEASURE`: Adds a measure block to the trigger model `trigger.READING_DIGITIZE`: Adds a digitize block to the trigger model

Details

This trigger model waits for a digital input or external trigger input event to occur, makes a measurement, and issues a notify event. If a digital output line is selected, a notify event asserts a digital output line. A notify event asserts the external output line regardless of the line settings. You can set the line to 7 to assert only the external output line, or to another setting to assert both a digital output line and the external output line.

Example

```
trigger.model.load("ExtDigTrigger", 7, 2, 10, 0.001, defbuffer1)
```

Set up the template to use the external in line and wait for a pulse from the external in to trigger measurements.
Pulse digital out line 2 and external out when the measurement is complete.
Make 10 measurements, with a delay of 1 ms before each measurement.
Store the measurements in `defbuffer1`.

Also see

None

trigger.model.load() — GradeBinning

This function loads a predefined trigger model configuration that sets up a grading operation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High, limit4Low)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High, limit4Low,
    limit4Pattern)
trigger.model.load("GradeBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High, limit4Low,
    limit4Pattern, bufferName)
```

components	The number of components to measure (1 to 268,435,455)
startInLine	The input line that starts the test; 5 for digital line 5, 6 for digital line 6, or 7 for external in; default is 5
startDelay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
endDelay	The delay time after the measurement (167 ns to 10,000 s); default is 0 for no delay
limitxHigh	x is limit 1, 2, 3, or 4; the upper limit that the measurement is compared against
limitxLow	x is 1, 2, 3, or 4; the lower limit that the measurement is compared against

limit1Pattern	The bit pattern that is sent when the measurement fails limit 1; range 1 to 15; default is 1
limit2Pattern	The bit pattern that is sent when the measurement fails limit 2; range 1 to 15; default is 2
limit3Pattern	The bit pattern that is sent when the measurement fails limit 3; range 1 to 15; default is 4
limit4Pattern	The bit pattern that is sent when the measurement fails limit 4; range 1 to 15; default is 8
allPattern	The bit pattern that is sent when all limits have passed; 1 to 15; default is 15
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; defaults to `defbuffer1`

Details

This trigger model template allows you to grade components and place them into up to four bins, based on the comparison to limits.

To set a limit as unused, set the high value for the limit to be less than the low limit.

All limit patterns and the pass pattern are sent on digital I/O lines 1 to 4, where 1 is the least significant bit.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

For a detailed example, see the section in the *Model DMM7510 User's Manual* named "Grading and binning resistors."

Also see

None

trigger.model.load() — Keithley2001

This function loads a predefined trigger model template that emulates a Keithley Instruments 2001 trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("Keithley2001", arm1Bypass, arm1Source, arm1Count, arm2Bypass,
    arm2Source, arm2Count, arm2Delay, trigBypass, trigSource, trigCount, trigDelay)
```

arm1Bypass	Bypass Arm 1: `trigger.ON` Do not bypass Arm 1: `trigger.OFF`
arm1Source	The event that triggers Arm 1; see **Details**
arm1Count	The number of times to repeat the Arm 1 layer
arm2Bypass	Bypass Arm 2: `trigger.ON` Do not bypass Arm 2: `trigger.OFF`
arm2Source	The event that triggers Arm 2; see **Details**
arm2Count	The number of times to repeat the Arm 2 layer
arm2Delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
trigBypass	Bypass the trigger layer: `trigger.ON` Do not bypass the trigger layer: `trigger.OFF`
trigSource	The event that triggers the trigger layer; see **Details**
trigCount	The number of times to repeat the trigger layer
trigDelay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay

Details

If the trigger layer is not bypassed, the external in/out rear-panel terminal is asserted. The arm layers do not assert the external in/out terminal.

You can use this template to emulate trigger models for products such as the Keithley Instruments Model 2000 and Model 2001 if you use only one of the arm layers. Set the other arm layers to a source of `trigger.EVENT_NONE`, a count of `1`, and a delay of `0` to simulate the immediate trigger option of a Model 2001 trigger model.

Trigger events	
Event description	**Event constant**
No trigger event (immediate)	`trigger.EVENT_NONE`
Front-panel TRIGGER key press (manual trigger)	`trigger.EVENT_DISPLAY`
A command interface trigger (bus trigger): • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.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 6)	`trigger.EVENT_DIGION`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

Refer to the application notes for the Model DMM7510 on the [Keithley Instruments website](http://www.keithley.com) *http://www.keithley.com* for an example with additional detail about this command.

Also see

None

trigger.model.load() — LoopUntilEvent

This function loads a predefined trigger model configuration that makes continuous measurements until the specified event occurs.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("LoopUntilEvent", triggerEvent, position)
trigger.model.load("LoopUntilEvent", triggerEvent, position, delay)
trigger.model.load("LoopUntilEvent", triggerEvent, position, delay, bufferName)
trigger.model.load("LoopUntilEvent", triggerEvent, position, delay, bufferName,
    readingBlock)
```

triggerEvent	The event that ends infinite triggering or readings set to occur before the trigger; see **Details**
position	The number of readings to make in relation to the size of the reading buffer; enter as a percentage (0 % to 100 %)
delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
bufferName	The name of the reading buffer, which may be a default buffer (`defbuffer1` or `defbuffer2`) or a user-defined buffer; defaults to `defbuffer1`
readingBlock	Define a measure or digitize block for the trigger model; options are: • `trigger.READING_ACTIVE`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `trigger.READING_ACTIVE` is used • `trigger.READING_MEASURE`: Adds a measure block to the trigger model • `trigger.READING_DIGITIZE`: Adds a digitize block to the trigger model

Details

The source event is the event that ends infinite triggering or ends readings set to occur before the trigger and start post-trigger readings. The trigger model makes readings until it detects the source event. After the source event, it makes a finite number of readings, based on the setting of the trigger position.

The position marks the location in the reading buffer where the trigger will occur. The position is set as a percentage of the active buffer capacity. The buffer captures measurements until a trigger occurs. When the trigger occurs, the buffer retains the percentage of readings specified by the position, then captures remaining readings until 100 percent of the buffer is filled. For example, if this is set to 75 for a reading buffer that holds 10,000 readings, the trigger model makes 2500 readings after it detects the source event. There will be 7500 pre-trigger readings and 2500 post-trigger readings.

The instrument makes two sets of readings. The first set is made until the trigger event occurs. The second set is made after the trigger event occurs, up to the number of readings calculated by the position parameter.

You cannot have the source event set at none when you run this predefined trigger model.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Trigger events	
Event description	**Event constant**
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender N (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer N (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

```
reset()

-- Set up measure function
dmm.measure.func = dmm.FUNC_DC_CURRENT

-- Initiate readings
trigger.model.load("LoopUntilEvent", trigger.EVENT_DISPLAY, 50)
trigger.model.initiate()
```

Reset the instrument.
Set the instrument to measure current.
Load the LoopUntilEvent trigger model to make measurements until the front panel trigger key is pressed, then continue to make measurements equal to 50 % of the reading buffer size.
Start the trigger model.

Also see

None

trigger.model.load() — SimpleLoop

This function loads a predefined trigger model configuration that does a specific number of measurements.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("SimpleLoop", count)
trigger.model.load("SimpleLoop", count, delay)
trigger.model.load("SimpleLoop", count, delay, bufferName)
trigger.model.load("SimpleLoop", count, delay, bufferName, readingBlock)
```

count	The number of measurements the instrument will make
delay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
bufferName	A string that indicates the reading buffer; the default buffers (`defbuffer1` or `defbuffer2`) or the name of a user-defined buffer; if no buffer is specified, `defbuffer1` is used
readingBlock	Define a measure or digitize block for the trigger model; options are: • `trigger.READING_ACTIVE`: Add a measure or digitize block to the trigger model based on the active function; if no option defined, `trigger.READING_ACTIVE` is used • `trigger.READING_MEASURE`: Adds a measure block to the trigger model • `trigger.READING_DIGITIZE`: Adds a digitize block to the trigger model

Details

This command sets up a loop that sets a delay, makes a measurement, and then repeats the loop the number of times you define in the count parameter.
The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

```
reset()

-- Set up measure function
dmm.measure.func = dmm.FUNC_DC_CURRENT
dmm.measure.autorange = dmm.ON
dmm.measure.nplc = 1

-- Initiate readings
trigger.model.load("SimpleLoop", 200)
trigger.model.initiate()
waitcomplete()

--Parse index and data into three columns
print("Rdg #", "Time (s)", "Current (A)")
for i = 1, defbuffer1.n do
      print(i, defbuffer1.relativetimestamps[i], defbuffer1[i])
end
```

This example uses the Simple Loop trigger model template to do a capacitor test. This example produces 200 readings that have output similar to the following example:

```
Rdg   #   Time (s)   Current (A)
1    0    -5.6898339156e-10
2    0.022129046   -5.6432783106e-10
3    0.063973966   -5.6329326206e-10
. . .
198   5.133657681   -5.5518916972e-10
199   5.155784187   -5.6363814801e-10
200   5.177910874   -5.6070686983e-10
```

Also see

None

trigger.model.load() — SortBinning

This function loads a predefined trigger model configuration that sets up a sorting operation.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern,  limit2High,
    limit2Low)trigger.model.load("SortBinning", components, startInLine, startDelay,
    endDelay, limit1High, limit1Low, limit1Pattern, allPattern, limit2High,
    limit2Low, limit2Pattern)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High, limit4Low)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High, limit4Low,
    limit4Pattern)
trigger.model.load("SortBinning", components, startInLine, startDelay, endDelay,
    limit1High, limit1Low, limit1Pattern, allPattern, limit2High, limit2Low,
    limit2Pattern, limit3High, limit3Low, limit3Pattern, limit4High, limit4Low,
    limit4Pattern, bufferName)
```

components	The number of components to measure (1 to 268,435,455)
limitxHigh	x is limit 1, 2, 3, or 4; the upper limit that the measurement is compared against
limitxLow	x is 1, 2, 3, or 4; the lower limit that the measurement is compared against
limit1Pattern	The bit pattern that is sent when the measurement passes limit 1; range 1 to 15; default is 1
limit2Pattern	The bit pattern that is sent when the measurement passes limit 2; range 1 to 15; default is 2
limit3Pattern	The bit pattern that is sent when the measurement passes limit 3; range 1 to 15; default is 4

limit4Pattern	The bit pattern that is sent when the measurement passes limit 4; range 1 to 15; default is 8
allPattern	The bit pattern that is sent when all limits have failed; 1 to 15; default is 15
startInLine	The input line that starts the test; 5 for digital line 5, 6 for digital line 6, or 7 for external in; default is 5
startDelay	The delay time before each measurement (167 ns to 10,000 s); default is 0 for no delay
endDelay	The delay time after the measurement (167 ns to 10,000 s); default is 0 for no delay
bufferName	The name of the reading buffer, which may be a default buffer (defbuffer1 or defbuffer2) or a user-defined buffer; defaults to defbuffer1

Details

This trigger model template allows you to sort components and place them into up to four bins, based on the comparison to limits.

To set a limit as unused, set the high value for the limit to be less than the low limit.

All limit patterns and the all fail pattern are sent on digital I/O lines 1 to 4, where 1 is the least significant bit.

The out line of the EXT TRIG IN/OUT rear-panel terminal is asserted at the end of each measurement.

Example

For a detailed example, see the section in the *Model DMM7510 User's Manual* named "Grading and binning resistors."

Also see

None

trigger.model.setblock() — trigger.BLOCK_BRANCH_ALWAYS

This function defines a trigger model block that always goes to a specific block.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_ALWAYS, branchToBlock)
```

blockNumber	The sequence of the block in the trigger model
branchToBlock	The block number to execute when the trigger model reaches the Always block

Details

When the trigger model reaches a branch-always building block, it goes to the building block set by *branchToBlock*.

Example

```
trigger.model.setblock(6, trigger.BLOCK_BRANCH_ALWAYS, 20)
```
When the trigger model reaches block 6, always branch to block 20.

Also see

None

trigger.model.setblock() — trigger.BLOCK_BRANCH_COUNTER

This function defines a trigger model block that branches to a specified block a specified number of times.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_COUNTER, targetCount,
    branchToBlock)
```

blockNumber	The sequence of the block in the trigger model
targetCount	The number of times to repeat
branchToBlock	The block number of the trigger model block to execute when the counter is less than the targetCount value

Details

This command defines a trigger model building block that branches to another block using a counter to iterate a specified number of times.

Counters increment every time the trigger model reaches them until they are more than or equal to the count value. At that point, the trigger model continues to the next building block in the sequence.

The counter is reset to 0 when the trigger model starts. It is incremented each time trigger model execution reaches the counter block.

If you are using remote commands, you can query the counter. The counter is incremented immediately before the branch compares the actual counter value to the set counter value. Therefore, the counter is at 0 until the first comparison. When the trigger model reaches the set counter value, branching stops and the counter value is one greater than the setting.

Example

```
trigger.model.setblock(4, trigger.BLOCK_BRANCH_COUNTER, 10, 2)
print(trigger.model.getbranchcount(4))
```
When the trigger model reaches this block, the trigger model returns to block 2. This repeats 10 times.
An example of the return if the trigger model has reached this block 5 times is:
5

Also see

trigger.model.getbranchcount() (on page 8-281)
trigger.model.setblock() — trigger.BLOCK_RESET_BRANCH_COUNT (on page 8-319)

trigger.model.setblock() — trigger.BLOCK_BRANCH_DELTA

This function defines a trigger model block that goes to a specified block if the difference of two measurements meets preset criteria.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_DELTA, targetDifference,
    branchToBlock)
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_DELTA, targetDifference,
    branchToBlock, measureBlock)
```

blockNumber	The sequence of the block in the trigger model
targetDifference	The value against which the block compares the difference between the measurements
branchToBlock	The block number of the trigger model block to execute when the difference between the measurements is less than or equal to the targetDifference
measureBlock	The block number of the measure or digitize block that makes the measurements to be compared; if this is 0 or undefined, the trigger model uses the previous measure or digitize block

Details

This block calculates the difference between the last two measurements from a measure or digitize block. It subtracts the most recent measurement from the previous measurement.

The difference between the measurements is compared to the target difference. If the difference is less than the target difference, the trigger model goes to the specified branching block. If the difference is more than the target difference, the trigger model proceeds to the next block in the trigger block sequence.

If you do not define the measure or digitize block, it will compare measurements of a measure or digitize block that precedes the branch delta block. For example, if you have a measure block, a wait block, another measure block, another wait block, and then the branch delta block, the delta block compares the measurements from the second measure block. If a preceding measure or digitize block does not exist, an error occurs.

Example

```
trigger.model.setblock(5, trigger.BLOCK_BRANCH_DELTA, 0.35, 8, 3)
```

Configure trigger block 5 to branch to block 8 when the measurement difference from block 3 is less than 0.35.

Also see

Delta building block (on page 3-84)

trigger.model.setblock() — trigger.BLOCK_BRANCH_LIMIT_CONSTANT

This function defines a trigger model block that branches to a block outside the normal trigger model flow if a measurement meets preset criteria.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_LIMIT_CONSTANT, limitType,
    limitA, limitB, branchToBlock)
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_LIMIT_CONSTANT, limitType,
    limitA, limitB, branchToBlock, measureBlock)
```

blockNumber	The sequence of the block in the trigger model
limitType	The type of limit, which can be one of the following types: • trigger.LIMIT_ABOVE • trigger.LIMIT_BELOW • trigger.LIMIT_INSIDE • trigger.LIMIT_OUTSIDE
limitA	The lower limit that the measurement is tested against; if limitType is set to: • trigger.LIMIT_ABOVE: This value is ignored • trigger.LIMIT_BELOW: The measurement must be below this value • trigger.LIMIT_INSIDE: The low limit that the measurement is compared against • trigger.LIMIT_OUTSIDE: The low limit that the measurement is compared against
limitB	The upper limit that the measurement is tested against; if limitType is set to: • trigger.LIMIT_ABOVE: The measurement must be above this value • trigger.LIMIT_BELOW: This value is ignored • trigger.LIMIT_INSIDE: The high limit that the measurement is compared against • trigger.LIMIT_OUTSIDE: The high limit that the measurement is compared against
branchToBlock	The block number of the trigger model block to execute when the measurement meets the defined criteria
measureBlock	The block number of the measure or digitize block that makes the measurements to be compared; if this is 0 or undefined, the trigger model uses the previous measure or digitize block

Details

The branch-on-constant-limits block goes to a branching block if a measurement meets the criteria set by this command.

The type of limit can be:

- Above: The measurement is above the value set by limit B; limit A must be set, but is ignored when this type is selected

- Below: The measurement is below the value set by limit A; limit B must be set, but is ignored when this type is selected

- Inside: The measurement is inside the values set by limits A and B; limit A must be the low value and Limit B must be the high value

- Outside: The measurement is outside the values set by limits A and B; limit A must be the low value and Limit B must be the high value

The measurement block must be a measure or digitize building block that occurs in the trigger model before the branch-on-constant-limits block. The last measurement from a measure or digitize building block is used.

If the limit A is more than the limit B, the values are automatically swapped so that the lesser value is used as the lower limit.

Example

```
trigger.model.setblock(5, trigger.BLOCK_BRANCH_LIMIT_CONSTANT, trigger.LIMIT_ABOVE,
    .1, 1, 2)
```

Sets trigger block 5 to be a constant limit that branches to block 2 when the measurement is above the value set for limit B (which is set to 1). Note that limit A must be set, but is ignored.

Also see

[Constant Limit building block](#) (on page 3-81)

trigger.model.setblock() — trigger.BLOCK_BRANCH_LIMIT_DYNAMIC

This function defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_LIMIT_DYNAMIC, limitType,
    limitNumber, branchToBlock)
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_LIMIT_DYNAMIC, limitType,
    limitNumber, branchToBlock, measureBlock)
```

blockNumber	The sequence of the block in the trigger model
limitType	The type of limit, which can be one of the following types: • trigger.LIMIT_ABOVE • trigger.LIMIT_BELOW • trigger.LIMIT_INSIDE • trigger.LIMIT_OUTSIDE
limitNumber	The limit number (1 or 2)
branchToBlock	The block number of the trigger model block to execute when the measurement meets the criteria set in the configuration list
measureBlock	The block number of the measure or digitize block that makes the measurements to be compared; if this is 0 or undefined, the trigger model uses the previous measure or digitize block

Details

The branch-on-dynamic-limits building block defines a trigger model block that goes to a specified block in the trigger model if a measurement meets user-defined criteria.

When you define this building block, you set:

- The type of limit (above, below, inside, or outside the limit values)
- The limit number (you can have 1 or 2 limits)
- The building block to go to if the measurement meets the criteria
- The building block that makes the measurement that is compared to the limits; the last measurement from that building block is used

There are two user-defined limits: limit 1 and limit 2. Both include their own high and low values, which are set using the front-panel Calculations limit settings or through commands. The results of these limit tests are recorded in the reading buffer that accompanies each stored reading.

Limit values are stored in the measure configuration list, so you can use a configuration list to step through different limit values.

The measure or digitize block must occur in the trigger model before the branch-on-dynamic-limits block. If no measure or digitize block is defined, the measurement from the previous measure or digitize block is used. If no previous measure or digitize block exists, an error is reported.

Example

```
trigger.model.setblock(7, trigger.BLOCK_BRANCH_LIMIT_DYNAMIC,
    trigger.LIMIT_OUTSIDE, 2, 10, 5)
```

Configure block 7 to check if limit 2 is outside its limit values, based on the measurements made in block 5. If values are outside the measurements, branch to block 10. If the values are not outside the measurements, trigger model execution continues to block 8.

Also see

Dynamic Limits building block (on page 3-83)
dmm.measure.limit[Y].low.value
dmm.measure.limit[Y].high.value

trigger.model.setblock() — trigger.BLOCK_BRANCH_ON_EVENT

This function branches to a specified block when a specified trigger event occurs.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_ON_EVENT, event,
    branchToBlock)
```

blockNumber	The sequence of the block in the trigger model
event	The event that must occur before the trigger model branches the specified block
branchToBlock	The block number of the trigger model block to execute when the specified event occurs

Details

The branch-on-event building block goes to a branching block after a specified trigger event occurs. If the trigger event has not yet occurred when the trigger model reaches the branch-on-event block, the trigger model continues to execute the blocks in the normal sequence. After the trigger event occurs, the next time the trigger model reaches the branch-on-event block, it goes to the branching block.

If you set the branch event to none, an error is generated when you run the trigger model.

The following table shows the constants for the events.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block *N* (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line *N* (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line *N* (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object *N* (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender *N* (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer *N* (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

```
trigger.model.setblock(6, trigger.BLOCK_BRANCH_ON_EVENT, trigger.EVENT_DISPLAY,
    2)
```
When the trigger model reaches this block, if the front-panel TRIGGER key has been pressed, the trigger model returns to block 2. If the TRIGGER key has not been pressed, the trigger model continues to block 7 (the next block in the trigger model).

Also see

trigger.model.setblock() — trigger.BLOCK_BRANCH_ONCE

This function causes the trigger model to branch to a specified building block the first time it is encountered in the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_ONCE, branchToBlock)
```

blockNumber	The sequence of the block in the trigger model
branchToBlock	The block number of the trigger model block to execute when the trigger model first encounters this block

Details

The branch-once building block branches to a specified block the first time trigger model execution encounters the branch-once block. If it is encountered again, the trigger model ignores the block and continues in the normal sequence.

The once block is reset when trigger model execution reaches the idle state. Therefore, the branch-once block always executes the first time the trigger model execution encounters this block.

Example

```
trigger.model.setblock(2, trigger.BLOCK_BRANCH_ONCE, 4)
```

When the trigger model reaches block 2, the trigger model goes to block 4 instead of going in the default sequence of block 3.

Also see

Once building block (on page 3-83)

trigger.model.setblock() — trigger.BLOCK_BRANCH_ONCE_EXCLUDED

This function defines a trigger model block that causes the trigger model to go to a specified building block every time the trigger model encounters it, except for the first time.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BRANCH_ONCE_EXCLUDED,
    branchToBlock)
```

blockNumber	The sequence of the block in the trigger model
branchToBlock	The block number of the trigger model block to execute when the trigger model encounters this block after the first encounter

Details

The branch-once-excluded building block is ignored the first time the trigger model encounters it. After the first encounter, the trigger model goes to the specified branching block.

The branch-once-excluded block is reset when the trigger model starts or is placed in idle.

Example

```
trigger.model.setblock(2, trigger.BLOCK_BRANCH_ONCE_EXCLUDED, 4)
```

When the trigger model reaches block 2 the first time, the trigger model goes to block 3. If the trigger model reaches this block again, the trigger model goes to block 4.

Also see

Once excluded building block (on page 3-84)

trigger.model.setblock() — trigger.BLOCK_BUFFER_CLEAR

This function defines a trigger model block that clears the reading buffer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_BUFFER_CLEAR)
trigger.model.setblock(blockNumber, trigger.BLOCK_BUFFER_CLEAR, bufferName)
```

blockNumber	The sequence of the block in the trigger model
bufferName	The name of the buffer, which must be an existing buffer; if no buffer is defined, defbuffer1 is used

Details

When trigger model execution reaches the buffer clear trigger block, the instrument empties the specified reading buffer. The specified buffer can be the default buffer or a buffer that you defined.

If you are clearing a user-defined reading buffer, you must create the buffer before you define this block.

Example

```
trigger.model.setblock(3, trigger.BLOCK_BUFFER_CLEAR, capTest2)
```

Assign trigger block 3 to buffer clear; when the trigger model reaches block 3, it clears the reading buffer named capTest2.

Also see

buffer.make() (on page 8-18)
Buffer clear building block (on page 3-77)

trigger.model.setblock() — trigger.BLOCK_CONFIG_NEXT

This function recalls the settings at the next index point of a configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_CONFIG_NEXT, configurationList)
```

blockNumber	The sequence of the block in the trigger model
configurationList	A string that defines the configuration list to recall

Details

When trigger model execution reaches a configuration recall next building block, the settings at the next index point in the specified configuration list are restored.

The first time the trigger model encounters this block for a specific configuration list, the first index point is recalled. Each subsequent time this block is encountered, the settings at the next index point in the configuration list are recalled and take effect before the next step executes. When the last index point in the list is reached, it returns to the first point.

The configuration list must be defined before you can use this building block.

Example

```
trigger.model.setblock(5, trigger.BLOCK_CONFIG_NEXT, "measTrigList")
```

Configure trigger block 5 to load the next index point in the configuration list named `measTrigList`.

Also see

Configuration lists (on page 3-32)

trigger.model.setblock() — trigger.BLOCK_CONFIG_PREV

This function defines a trigger model block that recalls the settings stored at the previous index point in a configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_CONFIG_PREV, configurationList)
```

blockNumber	The sequence of the block in the trigger model
configurationList	A string that defines the measure configuration list to recall

Details

The Config List Prev building block defines a trigger model block that recalls the settings stored at the previous index point in a configuration list.

The configuration list previous index trigger block type recalls the previous index point in a configuration list. It configures the settings of the instrument based on the settings at that index. The trigger model executes the settings at that index before the next block is executed.

The first time the trigger model encounters this block, the last index in the configuration list is recalled. Each subsequent time trigger model execution reaches a configuration list previous block for this configuration list, it goes backward one index. When the first index in the list is reached, it goes to the last index in the configuration list.

You must create the configuration list before you can define it in this building block.

Example

```
trigger.model.setblock(8, trigger.BLOCK_CONFIG_PREV, "measTrigList")
```
Configure trigger block 8 to load the previous index point in the configuration list named `measTrigList`.

Also see

Configuration lists (on page 3-32)

trigger.model.setblock() — trigger.BLOCK_CONFIG_RECALL

This function recalls the system settings that are stored in a configuration list.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_CONFIG_RECALL, configurationList)
trigger.model.setblock(blockNumber, trigger.BLOCK_CONFIG_RECALL, configurationList,
    index)
```

blockNumber	The sequence of the block in the trigger model
configurationList	A string that defines the configuration list to recall
index	The point in the configuration list to recall; default is 1

Details

When the trigger model reaches a configuration recall building block, the settings in the specified configuration list are recalled.

You can restore a specific set of configuration settings in the configuration list by defining the index.

Example

```
trigger.model.setblock(3, trigger.BLOCK_CONFIG_RECALL, "measTrigList", 5)
```
Configure trigger block 3 to load index point 5 from the configuration list named measTrigList.

Also see

Configuration lists (on page 3-32)

trigger.model.setblock() — trigger.BLOCK_DELAY_CONSTANT

This function adds a constant delay to the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_DELAY_CONSTANT, time)
```

blockNumber	The sequence of the block in the trigger model
time	The amount of time to delay in seconds (167 ns to 10,000 s, or 0 for no delay)

Details

When trigger model execution reaches a delay building block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made, and if any previously executed block started infinite measurements, they also continue to be made.

This delay waits for the set amount of delay time to elapse before proceeding to the next block in the trigger model.

If other delays have been set, this delay is in addition to the other delays.

Example

```
trigger.model.setblock(7, trigger.BLOCK_DELAY_CONSTANT, 30e-3)
```
Configure trigger block 7 to delay the trigger model before the next block until a delay of 30 ms elapses.

Also see

None

trigger.model.setblock() — trigger.BLOCK_DELAY_DYNAMIC

This function adds a delay to the execution of the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_DELAY_DYNAMIC,
    trigger.USER_DELAY_Mn)
```

blockNumber	The sequence of the block in the trigger model
n	The number of the user delay, 1 to 5, set by dmm.measure.userdelay[N] or dmm.digitize.userdelay[N]

Details

When trigger model execution reaches a dynamic delay building block, it stops normal measurement and trigger model operation for the amount of time set by the delay. Background measurements continue to be made.

Each measure and digitize function can have up to 5 unique user delay times (M1 to M5). The delay time is set by the user-delay command. This command is only available over a remote interface.

Example

```
dmm.measure.userdelay[1] = 5
trigger.model.setblock(1, trigger.BLOCK_DELAY_DYNAMIC, trigger.USER_DELAY_M1)
trigger.model.setblock(2, trigger.BLOCK_MEASURE)
trigger.model.setblock(3, trigger.BLOCK_BRANCH_COUNTER, 10, 1)
trigger.model.initiate()
```

Set user delay 1 for measurements to 5 s.
Set trigger block 1 to a dynamic delay that calls user delay 1.
Set trigger block 2 to make a measurement.
Set trigger block 3 to branch to block 1 ten times.
Start the trigger model.

Also see

dmm.digitize.userdelay[N] (on page 8-112)
dmm.measure.userdelay[N] (on page 8-194)

trigger.model.setblock() — trigger.BLOCK_DIGITAL_IO

This function defines a trigger model block that sets the lines on the digital I/O port high or low.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_DIGITAL_IO, bitPattern, bitMask)
```

blockNumber	The sequence of the block in the trigger model
bitPattern	Sets the value that specifies the output line bit pattern (0 to 63)
bitMask	Specifies the bit mask; if omitted, all lines are driven (0 to 63)

Details

To set the lines on the digital I/O port high or low, you can send a bit pattern. The pattern can be specified as a six-bit binary, hexadecimal, or integer value. The least significant bit maps to digital I/O line 1 and the most significant bit maps to digital I/O line 6.

The output line bit mask defines the bits in the pattern that are driven high or low. If the bit for a line is set to 1, the line is driven high. If the bit is set to 0, the line is driven low. A binary 1 in the bit mask indicates that the corresponding I/O line should be driven according to the bit pattern. To drive all lines, specify all ones (63, 0x3F, 0b111111) or omit this parameter.

For this building block to work as expected, make sure you configure the trigger type and line state of the digital line for use with the trigger model (use the digital line mode command).

Example

```
for x = 3,6 do digio.line[x].mode = digio.MODE_DIGITAL_OUT end
trigger.model.setblock(4, trigger.BLOCK_DIGITAL_IO, 20, 60)
```

The for loop configures digital I./O lines 3 through 6 as digital outputs. Trigger block 4 is then configured with a bit pattern of 20 (digital I/O lines 3 and 5 high). The optional bit mask is specified as 60 (lines 3 through 6), so both lines 3 and 5 are driven high.

Also see

digio.line[N].mode (on page 8-48)

trigger.model.setblock() — trigger.BLOCK_DIGITIZE

This function defines a trigger block that makes a measurement using a digitize function.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_DIGITIZE)
trigger.model.setblock(blockNumber, trigger.BLOCK_DIGITIZE, bufferName)
trigger.model.setblock(blockNumber, trigger.BLOCK_DIGITIZE, bufferName, count)
```

blockNumber	The sequence of the block in the trigger model
bufferName	The name of the buffer, which must be an existing buffer; if no buffer is defined, `defbuffer1` is used
count	The number of digitize readings to make before moving to the next block in the trigger model; set to a specific value or infinite (`trigger.COUNT_INFINITE`) or stop infinite (`trigger.COUNT_STOP`)

Details

When trigger model execution reaches the block:

1. The instrument begins making a measurement.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurement into the specified reading buffer.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and measurements continue in the background until the trigger model execution reaches another digitize block or until the trigger model ends.

A digitize function (digitize voltage or digitize current) must be selected before you run a trigger model that contains this block. You cannot have a measure block and a digitize block in the same trigger model.

Example

```
reset()
dmm.digitize.func = dmm.FUNC_DIGITIZE_VOLTAGE
dmm.digitize.samplerate = 50000
trigger.model.setblock(1, trigger.BLOCK_BUFFER_CLEAR, defbuffer1)
trigger.model.setblock(2, trigger.BLOCK_DELAY_CONSTANT, 0)
trigger.model.setblock(3, trigger.BLOCK_DIGITIZE, defbuffer1,
    trigger.COUNT_INFINITE)
trigger.model.setblock(4, trigger.BLOCK_WAIT, trigger.EVENT_DISPLAY)
trigger.model.setblock(5, trigger.BLOCK_DIGITIZE, defbuffer1,
    trigger.COUNT_STOP)
trigger.model.setblock(6, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFY1)
trigger.model.initiate()
waitcomplete()
print(defbuffer1.n)
```

Reset the instrument.
Set the function to digitize voltage.
Set block 1 to clear `defbuffer1`.
Set block 2 to set a delay of `0`.
Set block 3 to make digitize measurements infinitely.
Set block 4 to wait until the front-panel TRIGGER key is pressed.
Set block 5 to stop making digitize measurements.
Set block 6 to send a notification.
Start the trigger model.
Output the number of readings.

Also see

buffer.make() (on page 8-18)
Digitize building block (on page 3-76)

trigger.model.setblock() — trigger.BLOCK_LOG_EVENT

This function allows you to log an event in the event log when the trigger model is running.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_LOG_EVENT, eventNumber, message)
```

blockNumber	The sequence of the block in the trigger model
eventNumber	The event number: • `trigger.LOG_INFO`N • `trigger.LOG_WARN`N • `trigger.LOG_ERROR`N Where N is 1 to 4; you can define up to four of each type You can also set `trigger.LOG_WARN_ABORT`, which aborts the trigger model immediately and posts a warning event log message
message	A string up to 31 characters

Details

This block allows you to log an event in the event log when trigger model execution reaches this block. You can also force the trigger model to abort with this block. When the trigger model executes the block, the defined event is logged. If the abort option is selected, the trigger model is also aborted immediately.

You can define the type of event (information, warning, abort model, or error). All events generated by this block are logged in the event log. Warning and error events are also displayed in a popup on the front-panel display.

Note that using this block too often in a trigger model could overflow the event log. It may also take away from the time needed to process more critical trigger model blocks.

Example

```
trigger.model.setblock(9, trigger.BLOCK_LOG_EVENT, trigger.LOG_INFO2, "Trigger
    model complete.")
```

Set trigger model block 9 to log an event when the trigger model completes. In the event log, the message is:
`TM #1 block #9 logged: Trigger model complete.`

Also see

None

trigger.model.setblock() — trigger.BLOCK_MEASURE

This function defines a trigger block that makes a measurement.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_MEASURE)
trigger.model.setblock(blockNumber, trigger.BLOCK_MEASURE, bufferName)
trigger.model.setblock(blockNumber, trigger.BLOCK_MEASURE, bufferName, count)
```

blockNumber	The sequence of the block in the trigger model
bufferName	The name of the buffer, which must be an existing buffer; if no buffer is defined, defbuffer1 is used
count	Specifies the number of readings to make before moving to the next block in the trigger model; set to a specific value or to infinite by setting to trigger.COUNT_INFINITE; if not specified, defaults to 1

Details

When trigger model execution reaches the block:

1. The instrument begins making a measurement.
2. The trigger model execution waits for the measurement to complete.
3. The instrument places the measurement into the specified reading buffer.

If you are defining a user-defined reading buffer, you must create it before you define this block.

When you set the count to a finite value, trigger model execution remains at the block until all measurements are complete. If you set the count to infinite, the trigger model executes subsequent blocks and measurements continue in the background until the trigger model execution reaches another measure block or until the trigger model ends.

You must select a measure function before running a trigger model that contains this block.

You cannot include a measure block and a digitize block in the same trigger model.

Example

```
reset()
dmm.measure.func = dmm.FUNC_DC_VOLTAGE
trigger.model.setblock(1, trigger.BLOCK_BUFFER_CLEAR, defbuffer1)
trigger.model.setblock(2, trigger.BLOCK_DELAY_CONSTANT, 0)
trigger.model.setblock(3, trigger.BLOCK_MEASURE, defbuffer1,
    trigger.COUNT_INFINITE)
trigger.model.setblock(4, trigger.BLOCK_WAIT, trigger.EVENT_DISPLAY)
trigger.model.setblock(5, trigger.BLOCK_MEASURE, defbuffer1, trigger.COUNT_STOP)
trigger.model.setblock(6, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFY1)
trigger.model.initiate()
waitcomplete()
print(defbuffer1.n)
```

Reset the instrument.
Set the function to measure DC voltage.
Set block 1 to clear `defbuffer1`.
Set block 2 to set a delay of `0`.
Set block 3 to make measurements infinitely.
Set block 4 to wait until the front-panel TRIGGER key is pressed.
Set block 5 to stop making measurements.
Set block 6 to send a notification.
Start the trigger model.
You must press the front-panel TRIGGER key to stop measurements.
Output the number of readings.

Also see

buffer.make() (on page 8-18)
Measure building block (on page 3-75)

trigger.model.setblock() — trigger.BLOCK_NOP

This function creates a placeholder that performs no action in the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_NOP)
```

blockNumber	The sequence of the block in the trigger model

Details

If you remove a trigger model block, you can use this block as a placeholder for the block number so that you do not need to renumber the other blocks.

Example

`trigger.model.setblock(4, trigger.BLOCK_NOP)`	Set block number 4 to be a no operation block.

Also see

None

trigger.model.setblock() — trigger.BLOCK_NOTIFY

This function defines a trigger model block that generates a trigger event and immediately continues to the next block.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFYN)
```

blockNumber	The sequence of the block in the trigger model
N	The identification number of the notification; 1 to 8

Details

When trigger model execution reaches a notify block, the instrument generates a trigger event and immediately continues to the next block.

Other commands can reference the event that the notify block generates. This assigns a stimulus somewhere else in the system. For example, you can use the notify event as the stimulus of a hardware trigger line, such as a digital I/O line.

Example

```
digio.line[3].mode = digio.MODE_TRIGGER_OUT
trigger.model.setblock(5, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFY2)
trigger.digout[3].stimulus = trigger.EVENT_NOTIFY2
```
Define trigger model block 5 to be the notify 2 event. Assign the notify 2 event to be the stimulus for digital output line 3.

Also see

Notify building block (on page 3-80)

trigger.model.setblock() — trigger.BLOCK_RESET_BRANCH_COUNT

This function creates a block in the trigger model that resets a branch counter to 0.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_RESET_BRANCH_COUNT, counter)
```

blockNumber	The sequence of the block in the trigger model
counter	The block number of the counter that is to be reset

Details

When the trigger model reaches the Counter Reset block, it resets the count of the specified Branch on Counter block to zero.

Example

```
trigger.model.load("Empty")
trigger.model.setblock(1, trigger.BLOCK_BUFFER_CLEAR)
trigger.model.setblock(2, trigger.BLOCK_MEASURE)
trigger.model.setblock(3, trigger.BLOCK_BRANCH_COUNTER, 5, 2)
trigger.model.setblock(4, trigger.BLOCK_DELAY_CONSTANT, 1)
trigger.model.setblock(5, trigger.BLOCK_BRANCH_COUNTER, 3, 2)
trigger.model.setblock(6, trigger.BLOCK_RESET_BRANCH_COUNT, 3)
trigger.model.initiate()
waitcomplete()
print(defbuffer1.n)
```

Reset trigger model settings.
Clear `defbuffer1` at the beginning of the trigger model.
Loop and take 5 readings.
Delay a second.
Loop three more times back to block 2.
Reset block 3 to 0.
Start the trigger model and wait for measurements to complete.
Print the number of readings in the buffer.
Output:
`15`

Also see

trigger.model.getbranchcount() (on page 8-281)
trigger.model.setblock() — trigger.BLOCK_BRANCH_COUNTER (on page 8-297)

trigger.model.setblock() — trigger.BLOCK_WAIT

This function defines a trigger model block that waits for an event before allowing the trigger model to continue.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes	Restore configuration Instrument reset Power cycle	Configuration script	Not applicable

Usage

```
trigger.model.setblock(blockNumber, trigger.BLOCK_WAIT, event)
trigger.model.setblock(blockNumber, trigger.BLOCK_WAIT, event, logic, event)
trigger.model.setblock(blockNumber, trigger.BLOCK_WAIT, event, logic, event, event)
```

blockNumber	The sequence of the block in the trigger model
event	The event that must occur before the trigger block allows trigger execution to continue
logic	If each event must occur before the trigger model continues: `trigger.WAIT_AND` If at least one of the events must occur before the trigger model continues: `trigger.WAIT_OR`

Details

You can use the wait block to synchronize measurements with other instruments and devices.

You can set the instrument to wait for the following events:

* Front-panel TRIGGER key press
* Notify
* Command interface trigger
* Digital input/output signals, such as DB-9 and TSP-Link
* LAN
* Blender
* Timer
* Analog trigger
* External in trigger

The event can occur before trigger model execution reaches the wait block. If the event occurs after trigger model execution starts but before the trigger model execution reaches the wait block, the trigger model records the event. When trigger model execution reaches the wait block, it executes the wait block without waiting for the event to happen again.

The instrument clears the memory of the recorded event when trigger model execution is at the start block and when the trigger model exits the wait block.

All items in the list are subject to the same action; you cannot combine AND and OR logic in a single block.

You cannot leave the first event as no trigger. If the first event is not defined, the trigger model errors when you attempt to initiate it.

The following table shows the constants for the events.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block N (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line N (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line N (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object N (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender N (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer N (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

```
trigger.model.setblock(9, trigger.BLOCK_WAIT, trigger.EVENT_DISPLAY)
```
Set trigger model block 9 to wait for a user to press the TRIGGER key on the front panel before continuing.

Also see

trigger.model.state()

This function returns the present state of the trigger model.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
status, status, n = trigger.model.state()
```

status	The status of the trigger model:
	• trigger.STATE_IDLE
	• trigger.STATE_RUNNING
	• trigger.STATE_WAITING
	• trigger.STATE_EMPTY
	• trigger.STATE_BUILDING
	• trigger.STATE_FAILED
	• trigger.STATE_ABORTING
	• trigger.STATE_ABORTED
n	The last trigger model block that was executed

Details

This command returns the state of the trigger model. The instrument checks the state of a started trigger model every 100 ms.

This command returns the trigger state and the block that the trigger model last executed.

The trigger model states are:

- Idle: The trigger model is stopped.
- Running: The trigger model is running.
- Waiting: The trigger model has been in the same wait block for more than 100 ms.
- Empty: The trigger model is selected, but no blocks are defined.
- Building: Blocks have been added.
- Failed: The trigger model is stopped because of an error.
- Aborting: The trigger model is stopping because of a user request.
- Aborted: The trigger model is stopped because of a user request.

Example

print(trigger.model.state())	An example output if the trigger model is waiting and is at block 9 would be: trigger.STATE_WAITING trigger.STATE_WAITING 9

Also see

None

trigger.timer[N].clear()

This function clears the timer event detector and overrun indicator for the specified trigger timer number.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.timer[N].clear()
```

N	Trigger timer number (1 to 4)

Details

This command sets the timer event detector to the undetected state and resets the overrun indicator.

Example

`trigger.timer[1].clear()`	Clears trigger timer 1.

Also see

trigger.timer[N].count (on page 8-323)

trigger.timer[N].count

This attribute sets the number of events to generate each time the timer generates a trigger event or is enabled as a timer or alarm.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	1

Usage

```
count = trigger.timer[N].count
trigger.timer[N].count = count
```

count	Number of times to repeat the trigger (0 to 1,048,575)
N	Trigger timer number (1 to 4)

Details

If *count* is set to a number greater than 1, the timer automatically starts the next trigger timer delay at the expiration of the previous delay.

Set *count* to zero (0) to cause the timer to generate trigger events indefinitely.

If you use the trigger timer with a trigger model, make sure the count value is the same or more than any count values expected in the trigger model.

Example 1

`print(trigger.timer[1].count)`	Read trigger count for timer number 1.

Example 2

```
dmm.measure.nplc = 10
trigger.timer[4].reset()
trigger.timer[4].delay = 0.5
trigger.timer[4].start.stimulus = trigger.EVENT_NOTIFY8
trigger.timer[4].start.generate = trigger.OFF
trigger.timer[4].count = 20
trigger.timer[4].enable = trigger.ON

trigger.model.load("Empty")
trigger.model.setblock(1, trigger.BLOCK_BUFFER_CLEAR, defbuffer1)
trigger.model.setblock(2, trigger.BLOCK_NOTIFY, trigger.EVENT_NOTIFY8)
trigger.model.setblock(3, trigger.BLOCK_WAIT, trigger.EVENT_TIMER4)
trigger.model.setblock(4, trigger.BLOCK_MEASURE, defbuffer1)
trigger.model.setblock(5, trigger.BLOCK_BRANCH_COUNTER, 20, 3)
trigger.model.initiate()
waitcomplete()
print(defbuffer1.n)
```

Set NPLCs to 10.
Reset trigger timer 4.
Set trigger timer 4 to have a 0.5 s delay.
Set the stimulus for trigger timer 4 to be the notify 8 event.
Set the trigger timer 4 stimulus to off.
Set the timer event to occur when the timer delay elapses.
Set the trigger timer 4 count to 20.
Enable trigger timer 4.

Clear the trigger model.
Set trigger model block 1 to clear the buffer.
Set trigger model block 2 to generate the notify 8 event.
Set trigger model block 3 to wait for the trigger timer 4 to occur.
Set trigger model block 4 to make a measurement and store it in default buffer 1.
Set trigger model block 5 to repeat the trigger model 20 times, starting at block 3.
Start the trigger model.
Wait until all commands are complete.
Print the number of entries in default buffer 1.

Output:
`20`

Also see

trigger.timer[N].clear() (on page 8-323)
trigger.timer[N].delay (on page 8-325)
trigger.timer[N].reset() (on page 8-328)

trigger.timer[N].delay

This attribute sets and reads the timer delay.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	10e-6 (10 µs)

Usage

```
interval = trigger.timer[N].delay
trigger.timer[N].delay = interval
```

interval	Delay interval in seconds (8e-6 to 100,000)
N	Trigger timer number (1 to 4)

Details

Once the timer is enabled, each time the timer is triggered, it uses this delay period.

Assigning a value to this attribute is equivalent to:

```
trigger.timer[N].delaylist = {interval}
```

This creates a delay list of one value.

Reading this attribute returns the delay interval that will be used the next time the timer is triggered.

If you use the trigger timer with a trigger model, make sure the trigger timer delay is set so that the readings are paced correctly.

Example

`trigger.timer[1].delay = 50e-6`	Set the trigger timer 1 to delay for 50 µs.

Also see

trigger.timer[N].reset() (on page 8-328)

trigger.timer[N].delaylist

This attribute sets an array of timer intervals.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	{10e-6}

Usage

```
intervals = trigger.timer[N].delaylist
trigger.timer[N].delaylist = intervals
```

intervals	Table of delay intervals in seconds
N	Trigger timer number (1 to 4)

Details

Each time the timer is triggered after it is enabled, it uses the next delay period from the array. The default value is an array with one value of 10 µs.

After all elements in the array have been used, the delays restart at the beginning of the list.

If the array contains more than one element, the average of the delay intervals in the list must be ≥ 50 µs.

Example

| ```
trigger.timer[3].delaylist = {50e-6, 100e-6, 150e-6}

DelayList = trigger.timer[3].delaylist
for x = 1, table.getn(DelayList) do
   print(DelayList[x])
end
``` | Set a delay list on trigger timer 3 with three delays (50 µs, 100 µs, and 150 µs).<br><br>Read the delay list on trigger timer 3.<br><br>Output (assuming the delay list was set to 50 µs, 100 µs, and 150 µs):<br>`5.000000000e-05`<br>`1.000000000e-04`<br>`1.500000000e-04` |

Also see

[trigger.timer[N].reset()](#) (on page 8-328)

trigger.timer[N].enable

This attribute enables the trigger timer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	trigger.OFF

Usage

```
state = trigger.timer[N].enable
trigger.timer[N].enable = state
```

state	Disable the trigger timer: `trigger.OFF`
	Enable the trigger timer: `trigger.ON`
N	Trigger timer number (1 to 4)

Details

When this command is set to on, the timer performs the delay operation.

When this command is set to off, there is no timer on the delay operation.

You must enable a timer before it can use the delay settings or the alarm configuration. For expected results from the timer, it is best to disable the timer before changing a timer setting, such as delay or start seconds.

To use the timer as a simple delay or pulse generator with digital I/O lines, make sure the timer start time in seconds and fractional seconds is configured for a time in the past. To use the timer as an alarm, configure the timer start time in seconds and fractional seconds for the desired alarm time.

Example

`trigger.timer[3].enable = trigger.ON`	Enable the trigger timer for timer 3.

Also see

None

trigger.timer[N].reset()

This function resets trigger timer settings to their default values.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.timer[N].reset()
```

N	Trigger timer number (1 to 4)

Details

The `trigger.timer[N].reset()` function resets the following attributes to their default values:

- `trigger.timer[N].count`
- `trigger.timer[N].delay`
- `trigger.timer[N].delaylist`
- `trigger.timer[N].enable`
- `trigger.timer[N].start.generate`
- `trigger.timer[N].start.fractionalseconds`
- `trigger.timer[N].start.seconds`
- `trigger.timer[N].stimulus`

It also clears `trigger.timer[N].overrun`.

Example

`trigger.timer[1].reset()`	Resets the attributes associated with timer 1 to their default values.

Also see

trigger.timer[N].count (on page 8-323)
trigger.timer[N].delay (on page 8-325)
trigger.timer[N].delaylist (on page 8-325)
trigger.timer[N].enable (on page 8-327)
trigger.timer[N].start.fractionalseconds (on page 8-329)
trigger.timer[N].start.generate (on page 8-329)
trigger.timer[N].start.overrun (on page 8-330)
trigger.timer[N].start.seconds (on page 8-331)
trigger.timer[N].start.stimulus (on page 8-331)

trigger.timer[N].start.fractionalseconds

This attribute configures an alarm or a time in the future when the timer will start.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	0

Usage

```
time = trigger.timer[N].start.fractionalseconds
trigger.timer[N].start.fractionalseconds = time
```

time	The time in fractional seconds (0 to 2,147,483,647 ns)
N	Trigger timer number (1 to 4)

Details

This command configures the alarm of the timer.

When the timer is enabled, the timer starts immediately if the timer is configured for a start time that has passed.

Example

`trigger.timer[1].start.fractionalseconds = 40`	Set the trigger timer to start in 40 ns.

Also see

trigger.timer[N].start.generate

This attribute specifies when timer events are generated.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	trigger.OFF

Usage

```
state = trigger.timer[N].start.generate
trigger.timer[N].start.generate = state
```

state	Generate a timer event when the timer delay elapses: `trigger.OFF` Generate a timer event when the timer starts and when the delay elapses: `trigger.ON`
N	Trigger timer number (1 to 4)

Details

When this is set to on, a trigger event is generated immediately when the timer is triggered.

When it is set to off, a trigger event is generated when the timer elapses. This generates the event `trigger.EVENT_TIMERN`.

Example

```
trigger.timer[4].reset()
trigger.timer[4].delay = 0.5
trigger.timer[4].start.stimulus = trigger.EVENT_NOTIFY8
trigger.timer[4].start.generate = trigger.OFF
trigger.timer[4].count = 20
trigger.timer[4].enable = trigger.ON
```

Reset trigger timer 4.
Set trigger timer 4 to have a 0.5 s delay.
Set the stimulus for trigger timer 4 to be the notify 8 event.
Set the trigger timer 4 stimulus to off.
Set the trigger timer 4 count to 20.
Enable trigger timer 4.

Also see

trigger.timer[N].start.overrun

This attribute indicates if an event was ignored because of the event detector state.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (R)	Yes	Trigger timer *N* reset	Not applicable	Not applicable

Usage

```
state = trigger.timer[N].start.overrun
```

state	The trigger overrun state (`true` or `false`)
N	Trigger timer number (1 to 4)

Details

This command indicates if an event was ignored because the event detector was already in the detected state when the event occurred.

This is an indication of the state of the event detector built into the timer itself. It does not indicate if an overrun occurred in any other part of the trigger model or in any other construct that is monitoring the delay completion event. It also is not an indication of a delay overrun.

Example

`print(trigger.timer[1].start.overrun)`	If an event was ignored, the output is `true`. If the event was not ignored, the output is `false`.

Also see

trigger.timer[N].start.seconds

This attribute configures an alarm or a time in the future when the timer will start.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	0

Usage

```
time = trigger.timer[N].start.seconds
trigger.timer[N].start.seconds = time
```

time	The time: 0 to 2,147,483,647 s
N	Trigger timer number (1 to 4)

Details

This command configures the alarm of the timer.

When the timer is enabled, the timer starts immediately if the timer is configured for a start time that has passed.

Example

```
trigger.timer[1].start.seconds = localnode.gettime() + 30
trigger.timer[1].enable = trigger.ON
```

Set the trigger timer to start 30 s from the time when the timer is enabled.

Also see

None

trigger.timer[N].start.stimulus

This attribute describes the event that starts the trigger timer.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle Trigger timer *N* reset	Configuration script	trigger.EVENT_NONE

Usage

```
event = trigger.timer[N].start.stimulus
trigger.timer[N].start.stimulus = event
```

event	The event that starts the trigger timer
N	Trigger timer number (1 to 4)

Details

Set this attribute any trigger event to start the timer when that event occurs.

Set this attribute to zero (0) to disable event processing and use the timer as a timer or alarm based on the start time.

Trigger events are described in the table below.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block *N* (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: *`TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line *N* (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line *N* (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object *N* (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender *N* (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer *N* (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

`digio.line[3].mode = digio.MODE_TRIGGER_IN`
`trigger.timer[1].delay = 3e-3`
`trigger.timer[1].start.stimulus = trigger.EVENT_DIGIO3`
	Set digital I/O line 3 to be a trigger input.
	Set timer 1 to delay for 3 ms.
	Set timer 1 to start the timer when an event is detected on digital I/O line 3

Also see

None

trigger.timer[N].wait()

This function waits for a trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`triggered = trigger.timer[N].wait(timeout)`

`triggered`	Trigger detection indication
`N`	Trigger timer number (1 to 4)
`timeout`	Maximum amount of time in seconds to wait for the trigger

Details

If one or more trigger events were detected since the last time `trigger.timer[`*`N`*`].wait()` or `trigger.timer[`*`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 = trigger.timer[3].wait(10)` `print(triggered)`	Waits up to 10 s for a trigger on timer 3. If `false` is returned, no trigger was detected during the 10 s timeout. If `true` is returned, a trigger was detected.

Also see

[trigger.timer[N].clear()](#) (on page 8-323)

trigger.tsplinkin[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

`trigger.tsplinkin[`*`N`*`].clear()`	
`N`	The trigger line (1 to 3) to clear

Details

The trigger event detector enters the detected state when an event is detected. When this command is sent, the instrument does the following actions:

- Clears the trigger event detector
- Discards the history of the trigger line
- Clears the `trigger.tsplinkin[`*`N`*`].overrun` attribute

Example

`tsplink.line[2].mode =`
`    tsplink.MODE_TRIGGER_OPEN_DRAIN`
`trigger.tsplinkin[2].clear()`
	Clears the trigger event on TSP-Link line 2.

Also see

[trigger.tsplinkin[N].overrun](#) (on page 8-334)
[tsplink.line[N].mode](#) (on page 8-342)

trigger.tsplinkin[N].edge

This attribute indicates which trigger edge controls the trigger event detector for a trigger line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle TSP-Link line *N* reset	Configuration script	trigger.EDGE_FALLING

Usage

```
detectedEdge = trigger.tsplinkin[N].edge
trigger.tsplinkin[N].edge = detectedEdge
```

detectedEdge	The trigger mode: • Detect falling-edge triggers as inputs: `trigger.EDGE_FALLING` • Detect rising-edge triggers as inputs: `trigger.EDGE_RISING` • Detect either falling or rising-edge triggers as inputs: `trigger.EDGE_EITHER`
N	The trigger line (1 to 3)

Details

When the edge is detected, the instrument asserts a TTL-low pulse for the output.

The output state of the I/O line is controlled by the trigger logic. The user-specified output state of the line is ignored.

Example

`tsplink.line[3].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN` `trigger.tsplinkin[3].edge = trigger.EDGE_RISING`	Sets synchronization line 3 to detect rising edge triggers as input.

Also see

digio.writeport() (on page 8-52)
tsplink.line[N].mode (on page 8-342)
tsplink.line[N].reset() (on page 8-343)

trigger.tsplinkin[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 line *N* clear	Not applicable	Not applicable

Usage

```
overrun = trigger.tsplinkin[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(trigger.tsplinkin[1].overrun)`	If an event on line 1 was ignored, displays `true`; if no additional event occurred, displays `false`.

Also see

None

trigger.tsplinkin[N].wait()

This function waits for a trigger.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

`triggered = trigger.tsplinkin[N].wait(timeout)`	
`triggered`	Trigger detection indication; set to one of the following values: • `true`: A trigger is detected during the timeout period • `false`: A trigger is not detected during the timeout period
`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 are detected since the last time this command or `trigger.tsplinkin[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

`tsplink.line[3].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN`
`triggered = trigger.tsplinkin[3].wait(10)`
`print(triggered)`
	Waits up to 10 s for a trigger on TSP-Link® line 3.
	If `false` is returned, no trigger was detected during the 10-s timeout.
	If `true` is returned, a trigger was detected.

Also see

trigger.tsplinkin[N].clear() (on page 8-333)
tsplink.line[N].mode (on page 8-342)

trigger.tsplinkout[N].assert()

This function simulates the occurrence of the trigger and generates the corresponding trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

```
trigger.tsplinkout[N].assert()
```

N	The trigger line (1 to 3)

Details

Initiates a trigger event and does not wait for completion. The set pulse width determines how long the trigger is asserted.

Example

`tsplink.line[2].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN` `trigger.tsplinkout[2].assert()`	Asserts trigger on trigger line 2.

Also see

tsplink.line[N].mode (on page 8-342)

trigger.tsplinkout[N].logic

This attribute defines the trigger output with output logic for a trigger line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle TSP-Link line *N* reset	Configuration script	trigger.LOGIC_NEGATIVE

Usage

```
logicType = trigger.tsplinkout[N].logic
trigger.tsplinkout[N].logic = logicType
```

logicType	The output logic of the trigger generator: • Assert a TTL-high pulse for output: `trigger.LOGIC_POSITIVE` • Assert a TTL-low pulse for output: `trigger.LOGIC_NEGATIVE`
N	The trigger line (1 to 3)

Details

This attribute controls the logic that the output trigger generator uses on the given trigger line.

The output state of the digital I/O line is controlled by the trigger logic, and the user-specified output state of the line is ignored.

Example

`tsplink.line[3].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN` `trigger.tsplinkout[3].logic = trigger.LOGIC_POSITIVE`	Sets the trigger logic for synchronization line 3 to output a positive pulse.

Also see

[trigger.tsplinkout[N].assert()](#) (on page 8-336)
[tsplink.line[N].mode](#) (on page 8-342)

trigger.tsplinkout[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	Restore configuration Instrument reset Power cycle TSP-Link line *N* reset	Configuration script	10e-6 (10 μs)

Usage

```
width = trigger.tsplinkout[N].pulsewidth
trigger.tsplinkout[N].pulsewidth = width
```

width	The pulse width (0.0 to 100,000 s)
N	The trigger line (1 to 3)

Details

Setting the pulse width to 0 asserts the trigger indefinitely.

Example

`tsplink.line[3].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN` `trigger.tsplinkout[3].pulsewidth = 20e-6`	Sets pulse width for trigger line 3 to 20 μs.

Also see

[trigger.tsplinkout[N].assert()](#) (on page 8-336)
[trigger.tsplinkout[N].release()](#) (on page 8-338)
[tsplink.line[N].mode](#) (on page 8-342)

trigger.tsplinkout[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

```
trigger.tsplinkout[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.line[3].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN trigger.tsplinkout[3].release()```	Releases trigger line 3.

Also see

[trigger.tsplinkout[N].assert()](#) (on page 8-336)
[tsplink.line[N].mode](#) (on page 8-342)

trigger.tsplinkout[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	Restore configuration Instrument reset Power cycle TSP-Link line *N* reset	Configuration script	trigger.EVENT_NONE

Usage

```
event = trigger.tsplinkout[N].stimulus
trigger.tsplinkout[N].stimulus = event
```

event	The event identifier for the triggering event (see **Details**)
N	The trigger line (1 to 3)

Details

To disable automatic trigger assertion on the synchronization line, set this attribute to `trigger.EVENT_NONE`.

Do not use this attribute when triggering under script control. Use `trigger.tsplinkout[N].assert()` instead.

The *event* parameters that you can use are described in the table below.

Trigger events	
Event description	**Event constant**
No trigger event	`trigger.EVENT_NONE`
Front-panel TRIGGER key press	`trigger.EVENT_DISPLAY`
Notify trigger block *N* (1 to 8) generates a trigger event when the trigger model executes it	`trigger.EVENT_NOTIFYN`
A command interface trigger: • Any remote interface: `*TRG` • GPIB only: GET bus command • VXI-11: VXI-11 command `device_trigger`	`trigger.EVENT_COMMAND`
Line edge (either rising, falling, or either based on the configuration of the line) detected on digital input line *N* (1 to 6)	`trigger.EVENT_DIGION`
Line edge detected on TSP-Link synchronization line *N* (1 to 3)	`trigger.EVENT_TSPLINKN`
Appropriate LXI trigger packet is received on LAN trigger object *N* (1 to 8)	`trigger.EVENT_LANN`
Trigger event blender *N* (1 to 2), which combines trigger events	`trigger.EVENT_BLENDERN`
Trigger timer *N* (1 to 4) expired	`trigger.EVENT_TIMERN`
Analog trigger	`trigger.EVENT_ANALOGTRIGGER`
External in trigger	`trigger.EVENT_EXTERNAL`

Example

`print(trigger.tsplinkout[3].stimulus)`	Outputs the event that will start action on TSP-Link trigger line 3.

Also see

trigger.tsplinkout[N].assert() (on page 8-336)
tsplink.line[N].reset() (on page 8-343)

trigger.wait()

This function waits for a trigger event.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	Yes			

Usage

triggered = trigger.wait(*timeout*)

triggered	A trigger was detected during the timeout period: true No triggers were detected during the timeout period: false
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 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 s for a trigger.<br>If false is returned, no trigger was detected during the 10 s timeout.<br>If true is returned, a trigger was detected. |
|------|------|

Also see

[trigger.clear()](#) (on page 8-256)

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	Not applicable	Nonvolatile memory	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

[Using groups to manage nodes on a TSP-Link system](#) (on page 3-108)

tsplink.initialize()

This function initializes all instruments and enclosures in the TSP-Link system.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
nodesFound = tsplink.initialize()
tsplink.initialize()
tsplink.initialize(expectedNodes)
```

nodesFound	The number of nodes actually found on the system, including the node on which the command is running
expectedNodes	The number of nodes expected on the system (1 to 32)

Details

This function regenerates the system configuration information regarding the nodes connected to the TSP-Link system. You must initialize the system after making configuration changes. You need to initialize the system after you:

- Turn off power or reboot any instrument in the system

- Change node numbers on any instrument in the system

- Rearrange or disconnect the TSP-Link cable connections between instruments

If the only node on the TSP-Link network is the one running the command and *expectedNodes* is not provided, this function generates an error event. If you set *expectedNodes* to 1, the node is initialized.

If you include *expectedNodes*, if *nodesFound* is less than *expectedNodes*, an error event is generated.

Example

`nodesFound = tsplink.initialize(2)`
`print("Nodes found = " .. nodesFound)`
	Perform a TSP-Link initialization and indicate how many nodes are found.
	Example output if two nodes are found:
	`Nodes found = 2`
	Example output if fewer nodes are found and if `localnode.showevents = 7`:
	`1219, TSP-Link found fewer nodes than expected`
	`Nodes found = 1`

Also see

localnode.showevents (on page 8-224)
tsplink.node (on page 8-345)
tsplink.state (on page 8-346)

tsplink.line[N].mode

This attribute defines the trigger operation of a TSP-Link line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Restore configuration Instrument reset Power cycle TSP-Link line N reset	Configuration script	tsplink.MODE_DIGITAL_OPEN_DRAIN

Usage

```
mode = tsplink.line[N].mode
tsplink.line[N].mode = mode
```

`mode`	The trigger mode; see **Details**
`N`	The trigger line (1 to 3)

Details

This command defines whether or not the line is used as a digital or trigger control line and if it is an input or output.
The line mode can be set to the following options:

- TSP-Link digital open drain line: `tsplink.MODE_DIGITAL_OPEN_DRAIN`
- TSP-Link trigger open drain line: `tsplink.MODE_TRIGGER_OPEN_DRAIN`
- TSP-Link trigger synchronous master: `tsplink.MODE_SYNCHRONOUS_MASTER`
- TSP-Link trigger synchronous acceptor: `tsplink.MODE_SYNCHRONOUS_ACCEPTOR`

Example

`tsplink.line[3].mode = tsplink.MODE_TRIGGER_OPEN_DRAIN`	Sets the trigger mode for synchronization line 3 as a trigger open drain line.

Also see

trigger.tsplinkin[N].edge (on page 8-334)
trigger.tsplinkout[N].logic (on page 8-336)

tsplink.line[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.line[N].reset()
```

N	The trigger line (1 to 3)

Details

The `tsplink.line[N].reset()` function resets the following attributes to their default values:

- `trigger.tsplinkin[N].edge`
- `trigger.tsplinkout[N].logic`
- `tsplink.line[N].mode`
- `trigger.tsplinkout[N].stimulus`
- `trigger.tsplinkout[N].pulsewidth`

This also clears `trigger.tsplinkin[N].overrun`.

Example

`tsplink.line[3].reset()`	Resets TSP-Link trigger line 3 attributes to default values.

Also see

tsplink.line[N].state

This attribute reads or writes the digital state of a TSP-Link synchronization line.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	Yes	Not applicable	Nonvolatile memory	tsplink.STATE_HIGH

Usage

```
lineState = tsplink.line[N].state
tsplink.line[N].state = lineState
```

lineState	The state of the synchronization line: • Low: `tsplink.STATE_LOW` • High: `tsplink.STATE_HIGH`
N	The trigger line (1 to 3)

Details

Use `tsplink.writeport()` to write to all TSP-Link synchronization lines.

The reset function does not affect the present states of the TSP-Link trigger lines.

Example

`lineState = tsplink.line[3].state` `print(lineState)`	Assume line 3 is set high, and then the state is read. Output: `tsplink.STATE_HIGH`

Also see

tsplink.line[N].mode (on page 8-342)
tsplink.writeport() (on page 8-346)

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 (1 to 64)

Details

This attribute returns the node number of the master in a set of instruments connected using TSP-Link.

Example

`LinkMaster = tsplink.master`	Store the TSP-Link master node number in a variable called LinkMaster.

Also see

tsplink.initialize() (on page 8-341)

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	2

Usage

```
nodeNumber = tsplink.node
tsplink.node = nodeNumber
```

nodeNumber	The node number of the instrument or enclosure (1 to 64)

Details

This attribute 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 or `tsplink.initialize()` 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.initialize() (on page 8-341)
tsplink.state (on page 8-346)

tsplink.readport()

This function reads the TSP-Link synchronization 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

Triggering using TSP-Link synchronization lines (on page 3-107)
tsplink.line[N].state (on page 8-343)
tsplink.writeport() (on page 8-346)

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.initialize()` or `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 8-345)

tsplink.writeport()

This function writes to all TSP-Link synchronization lines as a digital I/O port.

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).
>
> 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.line[N].state](#) (on page 8-343)

tspnet.clear()

This function clears any pending output data from the instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

> `tspnet.clear(connectionID)`
>
> | | |
> |---|---|
> | `connectionID` | The connection ID returned from `tspnet.connect()` |

Details

> This function clears any pending output data from the device. No data is returned to the caller and no data is processed.

Example

`tspnet.write(testdevice, "print([[hello]])")`
`print(tspnet.readavailable(testdevice))`
 	
`tspnet.clear(testdevice)`
`print(tspnet.readavailable(testdevice))`
	Write data to a device, then print how much is available.
	Output:
	`6.00000e+00`
	
	Clear data and print how much data is available again.
	Output:
	`0.00000e+00`

Also see

> [tspnet.connect()](#) (on page 8-348)
> [tspnet.readavailable()](#) (on page 8-352)
> [tspnet.write()](#) (on page 8-357)

tspnet.connect()

This function establishes a network connection with another LAN instrument or device through the LAN interface.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
connectionID = tspnet.connect(ipAddress)
connectionID = tspnet.connect(ipAddress, portNumber, initString)
```

connectionID	The connection ID to be used as a handle in all other tspnet function calls
ipAddress	IP address to which to connect in a string
portNumber	Port number (default 5025)
initString	Initialization string to send to ipAddress

Details

This command connects a device to another device through the LAN interface. If the *portNumber* is 23, the interface uses the Telnet protocol and sets appropriate termination characters to communicate with the device.

If a *portNumber* and *initString* are provided, it is assumed that the remote device is not TSP-enabled. The Model DMM7510 does not perform any extra processing, prompt handling, error handling, or sending of commands. In addition, the `tspnet.tsp.*` commands cannot be used on devices that are not TSP-enabled.

If neither a *portNumber* nor an *initString* is provided, the remote device is assumed to be a Keithley Instruments TSP-enabled device. Depending on the state of the `tspnet.tsp.abortonconnect` attribute, the Model DMM7510 sends an `abort` command to the remote device on connection.

You can simultaneously connect to a maximum of 32 remote devices.

Example 1

```
instrumentID = tspnet.connect("192.0.2.1")
if instrumentID then
   -- Use instrumentID as needed here
   tspnet.disconnect(instrumentID)
end
```
Connect to a TSP-enabled device.

Example 2

```
instrumentID = tspnet.connect("192.0.2.1", 1394, "*rst\r\n")
if instrumentID then
   -- Use instrumentID as needed here
   tspnet.disconnect(instrumentID)
end
```
Connect to a device that is not TSP-enabled.

Also see

tspnet.disconnect()

This function disconnects a specified TSP-Net session.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
tspnet.disconnect(connectionID)
```

connectionID	The connection ID returned from tspnet.connect()

Details

This function disconnects the two devices by closing the connection. The *connectionID* is the session handle returned by tspnet.connect().

For TSP-enabled devices, this aborts any remotely running commands or scripts.

Example

`testID = tspnet.connect("192.0.2.0")`
`-- Use the connection`
`tspnet.disconnect(testID)`
	Create a TSP-Net session.
	
	Close the session.

Also see

tspnet.execute()

This function sends a command string to the remote device.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
tspnet.execute(connectionID, commandString)
value1 = tspnet.execute(connectionID, commandString, formatString)
value1, value2 = tspnet.execute(connectionID, commandString, formatString)
value1, ..., valueN = tspnet.execute(connectionID, commandString, formatString)
```

connectionID	The connection ID returned from tspnet.connect()
commandString	The command to send to the remote device
value1	The first value decoded from the response message
value2	The second value decoded from the response message
valueN	The Nth value decoded from the response message; there is one return value for each format specifier in the format string
...	One or more values separated with commas
formatString	Format string for the output

Details

This command sends a command string to the remote instrument. A termination is added to the command string when it is sent to the remote instrument (`tspnet.termination()`). You can also specify a format string, which causes the command to wait for a response from the remote instrument. The Model DMM7510 decodes the response message according to the format specified in the format string and returns the message as return values from the function (see `tspnet.read()` for format specifiers).

When this command is sent to a TSP-enabled instrument, the Model DMM7510 suspends operation until a timeout error is generated or until the instrument responds. The TSP prompt from the remote instrument is read and discarded. The Model DMM7510 places any remotely generated errors and events into its event queue. When the optional format string is not specified, this command is equivalent to `tspnet.write()`, except that a termination is automatically added to the end of the command.

Example 1

`tspnet.execute(instrumentID, "runScript()")`	Command the remote device to run a script named `runScript`.

Example 2

`tspnet.termination(instrumentID, tspnet.TERM_CRLF)`
`tspnet.execute(instrumentID, "*idn?")`
`print("tspnet.execute returns:", tspnet.read(instrumentID))`
	Print the `*idn?` string from the remote device.

Also see

tspnet.idn()

This function retrieves the response of the remote device to `*IDN?`.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

`idnString = tspnet.idn(connectionID)`

`idnString`	The returned `*IDN?` string
`connectionID`	The connection ID returned from `tspnet.connect()`

Details

This function retrieves the response of the remote device to `*IDN?`.

Example

`deviceID = tspnet.connect("192.0.2.1")`
`print(tspnet.idn(deviceID))`
`tspnet.disconnect(deviceID)`
	Assume the instrument is at IP address 192.0.2.1.
	The output that is produced when you connect to the instrument and read the IDN string may appear as:
	`KEITHLEY INSTRUMENTS,MODEL`
	`DMM7510,00000170,1.0.0a`

Also see

tspnet.read()

This function reads data from a remote device.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
value1 = tspnet.read(connectionID)
value1 = tspnet.read(connectionID, formatString)
value1, value2 = tspnet.read(connectionID, formatString)
value1, ..., valueN = tspnet.read(connectionID, formatString)
```

value1	The first value decoded from the response message
value2	The second value decoded from the response message
valueN	The nth value decoded from the response message; there is one return value for each format specifier in the format string
...	One or more values separated with commas
connectionID	The connection ID returned from `tspnet.connect()`
formatString	Format string for the output, maximum of 10 specifiers

Details

This command reads available data from the remote instrument and returns responses for the specified number of arguments.

The format string can contain the following specifiers:

`%[width]s`	Read data until the specified length
`%[max width]t`	Read data until the specified length or until punctuation is found, whichever comes first
`%[max width]n`	Read data until a newline or carriage return
`%d`	Read a number (delimited by punctuation)

A maximum of 10 format specifiers can be used for a maximum of 10 return values.

If *formatString* is not provided, the command returns a string that contains the data until a new line is reached. If no data is available, the Model DMM7510 pauses operation until the requested data is available or until a timeout error is generated. Use `tspnet.timeout` to specify the timeout period.

When the Model DMM7510 reads from a TSP-enabled remote instrument, the Model DMM7510 removes Test Script Processor (TSP®) prompts and places any errors or events it receives from the remote instrument into its own event queue. The Model DMM7510 prefaces events and errors from the remote device with `Remote Error`, followed by the event number and description.

Example

`tspnet.write(deviceID, "*idn?\r\n")`
 	
`print("write/read returns:", tspnet.read(deviceID))`
	Send the `"*idn?\r\n"` message to the instrument connected as `deviceID`.
	Display the response that is read from `deviceID` (based on the `*idn?` message).

Also see

tspnet.connect() (on page 8-348)
tspnet.readavailable() (on page 8-352)
tspnet.timeout (on page 8-354)
tspnet.write() (on page 8-357)

tspnet.readavailable()

This function checks to see if data is available from the remote device.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

`bytesAvailable = tspnet.readavailable(connectionID)`

`bytesAvailable`	The number of bytes available to be read from the connection
`connectionID`	The connection ID returned from `tspnet.connect()`

Details

This command checks to see if any output data is available from the device. No data is read from the instrument. This allows TSP scripts to continue to run without waiting on a remote command to finish.

Example

`ID = tspnet.connect("192.0.2.1")`
`tspnet.write(ID, "*idn?\r\n")`
 	
`repeat bytes = tspnet.readavailable(ID) until bytes > 0`
 	
`print(tspnet.read(ID))`
`tspnet.disconnect(ID)`
	Send commands that will create data.
	
	Wait for data to be available.

Also see

tspnet.connect() (on page 8-348)
tspnet.read() (on page 8-351)

tspnet.reset()

This function disconnects all TSP-Net sessions.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
tspnet.reset()
```

Details

This command disconnects all remote instruments connected through TSP-Net. For TSP-enabled devices, this causes any commands or scripts running remotely to be terminated.

Also see

None

tspnet.termination()

This function sets the device line termination sequence.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
type = tspnet.termination(connectionID)
type = tspnet.termination(connectionID, termSequence)
```

type	The termination type: • `tspnet.TERM_LF` • `tspnet.TERM_CR` • `tspnet.TERM_CRLF` • `tspnet.TERM_LFCR`
connectionID	The connection ID returned from `tspnet.connect()`
termSequence	The termination sequence: • `tspnet.TERM_LF` • `tspnet.TERM_CR` • `tspnet.TERM_CRLF` • `tspnet.TERM_LFCR`

Details

This function sets and gets the termination character sequence that is used to indicate the end of a line for a TSP-Net connection.

Using the *termSequence* parameter sets the termination sequence. The present termination sequence is always returned.

For the *termSequence* parameter, use the same values listed in the table above for type. There are four possible combinations, all of which are made up of line feeds (LF or 0x10) and carriage returns (CR or 0x13). For TSP-enabled devices, the default is `tspnet.TERM_LF`. For devices that are not TSP-enabled, the default is `tspnet.TERM_CRLF`.

Example

| ```
deviceID = tspnet.connect("192.0.2.1")
if deviceID then
    tspnet.termination(deviceID, tspnet.TERM_LF)
end
``` | Sets termination type for IP address 192.0.2.1 to `TERM_LF`. |

Also see

tspnet.timeout

This attribute sets the timeout value for the `tspnet.connect()`, `tspnet.execute()`, and `tspnet.read()` commands.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Restore configuration Instrument reset Power cycle	Configuration script	20.0 (20 s)

Usage

```
value = tspnet.timeout
tspnet.timeout = value
```

value	The timeout duration in seconds (0.001 to 30.0 s)

Details

This attribute sets the amount of time the `tspnet.connect()`, `tspnet.execute()`, and `tspnet.read()` commands will wait for a response.

The time is specified in seconds. The timeout may be specified to millisecond resolution, but is only accurate to the nearest 10 ms.

Example

`tspnet.timeout = 2.0`	Sets the timeout duration to 2 s.

Also see

tspnet.tsp.abort()

This function causes the TSP-enabled instrument to stop executing any of the commands that were sent to it.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
tspnet.tsp.abort(connectionID)
```

connectionID	Integer value used as a handle for other `tspnet` commands

Details

This function is appropriate only for TSP-enabled instruments.

Sends an abort command to the remote instrument.

Example

`tspnet.tsp.abort(testConnection)`	Stops remote instrument execution on `testConnection`.

Also see

None

tspnet.tsp.abortonconnect

This attribute contains the setting for abort on connect to a TSP-enabled instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Attribute (RW)	No	Restore configuration Instrument reset Power cycle	Configuration script	1 (enable)

Usage

```
tspnet.tsp.abortonconnect = value
value = tspnet.tsp.abortonconnect
```

value	• Enable: `1` • Disable: `0`

Details

This setting determines if the instrument sends an abort message when it attempts to connect to a TSP-enabled instrument using the `tspnet.connect()` function.

When you send the abort command on an interface, it causes any other active interface on that instrument to close. If you do not send an abort command (or if `tspnet.tsp.abortonconnect` is set to 0) and another interface is active, connecting to a TSP-enabled remote instrument results in a connection. However, the instrument will not respond to subsequent reads or executes because control of the instrument is not obtained until an abort command has been sent.

Example

`tspnet.tsp.abortonconnect = 0`	Configure the instrument so that it does not send an abort command when connecting to a TSP-enabled instrument.

Also see

tspnet.tsp.rbtablecopy()

This function copies a reading buffer synchronous table from a remote instrument to a TSP-enabled instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
table = tspnet.tsp.rbtablecopy(connectionID, name)
table = tspnet.tsp.rbtablecopy(connectionID, name, startIndex, endIndex)
```

table	A copy of the synchronous table or a string
connectionID	Integer value used as a handle for other `tspnet` commands
name	The full name of the reading buffer name and synchronous table to copy
startIndex	Integer start value
endIndex	Integer end value

Details

This function is only appropriate for TSP-enabled instruments.

This function reads the data from a reading buffer on a remote instrument and returns an array of numbers or a string representing the data. The *startIndex* and *endIndex* parameters specify the portion of the reading buffer to read. If no index is specified, the entire buffer is copied.

The function returns a table if the table is an array of numbers; otherwise a comma-delimited string is returned.

This command is limited to transferring 50,000 readings at a time.

Example

| ```
times =
      tspnet.tsp.rbtablecopy(testTspdevice,
   "testRemotebuffername.timestamps", 1, 3)
print(times)
``` | Copy the specified timestamps table for items 1 through 3, then display the table. Example output:<br>`01/01/2011`<br>`    10:10:10.0000013,01/01/2011`<br>`    10:10:10.0000233,01/01/2011`<br>`    10:10:10.0000576` |

Also see

None

tspnet.tsp.runscript()

This function loads and runs a script on a remote TSP-enabled instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
tspnet.tsp.runscript(connectionID, name, script)
```

connectionID	Integer value used as an identifier for other tspnet commands
name	The name that is assigned to the script
script	The body of the script as a string

Details

This function is appropriate only for TSP-enabled instruments.

This function downloads a script to a remote instrument and runs it. It automatically adds the appropriate loadscript and endscript commands around the script, captures any errors, and reads back any prompts. No additional substitutions are done on the text.

The script is automatically loaded, compiled, and run.

Any output from previous commands is discarded.

This command does not wait for the script to complete.

If you do not want the script to do anything immediately, make sure the script only defines functions for later use. Use the tspnet.execute() function to execute those functions at a later time.

Example

```
tspnet.tsp.runscript(myConnection, "myTest",
"print([[start]]) for d = 1, 10 do print([[work]]) end print([[end]])")
```
Load and run a script entitled myTest on the TSP-enabled instrument connected with myConnection.

Also see

tspnet.execute() (on page 8-349)

tspnet.write()

This function writes a string to the remote instrument.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
tspnet.write(connectionID, inputString)
```

connectionID	The connection ID returned from tspnet.connect()
inputString	The string to be written

Details

The `tspnet.write()` function sends *inputString* to the remote instrument. It does not wait for command completion on the remote instrument.

The Model DMM7510 sends *inputString* to the remote instrument exactly as indicated. The *inputString* must contain any necessary new lines, termination, or other syntax elements needed to complete properly.

Because `tspnet.write()` does not process output from the remote instrument, do not send commands that generate too much output without processing the output. This command can stop executing if there is too much unprocessed output from previous commands.

Example

`tspnet.write(myID, "runscript()\r\n")`	Commands the remote instrument to execute a command or script named `runscript()` on a remote device identified in the system as `myID`.

Also see

[tspnet.connect()](#) (on page 8-348)
[tspnet.read()](#) (on page 8-351)

upgrade.previous()

This function returns to a previous version of the Model DMM7510 firmware.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
upgrade.previous()
```

Details

This function allows you to revert to an earlier version of the firmware.

When you send this function, the instrument searches the USB flash drive that is inserted in the front-panel USB port for an upgrade file. If the file is found, the instrument performs the upgrade. An error is returned if an upgrade file is not found.

NOTE

Use this command with caution. Make sure your instrument can support the earlier version and that there are no compatibility issues. Check with Keithley Instruments before using this command if you have questions.

Also see

[Upgrading the firmware](#) (on page A-4)
[upgrade.unit()](#) (on page 8-359)

upgrade.unit()

This function upgrades the Model DMM7510 firmware.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
upgrade.unit()
```

Details

When `upgrade.unit()` is used, the firmware is only loaded if the version of the firmware is newer than the existing version. If the version is older or at the same revision level, it is not upgraded.

When you send this function, the instrument searches the USB flash drive that is inserted in the front-panel USB port for an upgrade file. If the file is found, the instrument verifies that the file is a newer version. If the version is older or at the same revision level, it is not upgraded. If it is a newer version, the instrument performs the upgrade. An error event message is returned if no upgrade file is found.

Also see

upgrade.previous() (on page 8-358)
Upgrading the firmware (on page A-4)

userstring.add()

This function adds a user-defined 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.

Also see

userstring.catalog() (on page 8-360)
userstring.delete() (on page 8-361)
userstring.get() (on page 8-361)

userstring.catalog()

This function creates an iterator for the user-defined string catalog.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
for name in userstring.catalog() do body end
```

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.catalog() do```
```   userstring.delete(name)```
```end```
	Deletes all user-defined strings in nonvolatile memory.

Example 2

```for name in userstring.catalog() do```
```   print(name .. " = " ..```
```   userstring.get(name))```
```end```
	Prints all `userstring` key-value pairs.
	Output:
	`product = Widgets`
	`assetnumber = 236`
	`contact = John Doe`
	The above output lists the user-defined strings added in the example for the `userstring.add()` function. Notice the key-value pairs are not listed in the order they were added.

Also see

userstring.add() (on page 8-359)
userstring.delete() (on page 8-361)
userstring.get() (on page 8-361)

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 8-359)
userstring.catalog() (on page 8-360)
userstring.get() (on page 8-361)

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

`value = userstring.get("assetnumber")` `print(value)`	Read the value associated with a 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 8-359)
userstring.catalog() (on page 8-360)
userstring.delete() (on page 8-361)

waitcomplete()

This function waits for all overlapped commands to complete.

Type	TSP-Link accessible	Affected by	Where saved	Default value
Function	No			

Usage

```
waitcomplete()
waitcomplete(group)
```

`group`	Specifies which TSP-Link group on which to wait

Details

This function will wait for all previously started overlapped commands to complete.

A group number may only be specified when this node is the master node.

If no `group` is specified, the local group is used.

If zero (0) is specified for the `group`, this function waits for all nodes in the system.

> ## NOTE
> Any nodes that are not assigned to a group (group number is 0) are part of the master node's group.

Example 1

`waitcomplete()`	Waits for all nodes in the local group.

Example 2

`waitcomplete(G)`	Waits for all nodes in group G.

Example 3

`waitcomplete(0)`	Waits for all nodes on the TSP-Link network.

Also see

None

Frequently asked questions (FAQs)

In this section:

I see a command that is not listed in the Command Reference section. What is it?

You may see commands that are internal to the instrument if you:

- Have the event log set to record commands
- Capture commands with the create setup feature
- Capture commands with the record macro feature
- Store settings in a configuration list

If a command is not described in the Command Reference section, do not use it.

The commands that you might see include:

- `dmm.measure.configlist.set()`
- `dmm.performance`

How do I display the instrument's serial number?

The instrument serial number is on a label on the rear panel of the instrument. You can also access the serial number from the front panel using the front-panel keys and menus.

To view the system information from the front panel:
1. Press the **MENU** key.
2. Under System, select **Information**. The system information displays, including the serial number.
3. To return to the Home screen, select the **HOME** key.

To view system information using SCPI commands:

Send the command:

```
*IDN?
```

To view system information using TSP commands:

Send the command:

```
print(localnode.serialno)
```

What VISA resource name is required?

To determine the VISA resource name that is required to communicate with the instrument, you can run the Keithley Configuration Panel. The Configuration Panel automatically detects all instruments connected to the computer.

If you installed the Keithley I/O Layer, you can access the Keithley Configuration Panel through the Microsoft® Windows® Start menu.

To run the Configuration Panel, click **Start > All Programs > Keithley Instruments > Keithley Configuration Panel** and follow the steps in the wizard.

Can I use Agilent GPIB cards with Keithley drivers?

Yes, as long as the instrument driver uses VISA for instrument communication. This is true for any instrument driver that is IVI or VXI/PnP based.

How do I check the USB driver for the device?

To check the driver for the USB Test and Measurement Device:
1. Open Device Manager.

♀ Quick Tip

From the Start menu, you can enter `Devmgmt.msc` in the Run box or the Windows 7 search box to start Device Manager.

2. Under USB Test and Measurement Devices, look for USB Test and Measurement Device.

   If the device is not there, either VISA is not installed or the instrument is not plugged in and switched on.

Figure 162: Device Manager dialog box showing USB Test and Measurement Device

3. Right-click the device.
4. Select Properties.
5. Select the Driver tab.
6. Click **Driver Details**.
7. Verify that the device driver is the winusb.sys. driver from Microsoft.

Figure 163: Driver File Details dialog box

8. If the incorrect driver is installed, click **OK**.
9. On the Driver tab, click **Update Driver**.
10. Browse for the driver; select the `C:\windows\inf` folder. Locate the `winusb.inf` file. Select this and make sure the driver is now in use.
11. If this does not work, uninstall VISA, unplug the instrument and follow the steps to reinstall VISA in the section <u>Modifying, repairing, or removing Keithley I/O Layer software</u> (on page 2-85).

Which Microsoft Windows operating systems are supported?

Microsoft Windows 2000, Windows XP, Windows Vista, and Windows 7 are supported.

What to do if the GPIB controller is not recognized?

If the hardware is not recognized by the computer:

1. Uninstall the software drivers.
2. Reboot the computer.
3. Check for newer drivers on the vendor's website. Check that the drivers are valid for the operating system you have and any updates that might be necessary. This information is typically found in the readme file that comes with the drivers.
4. Install software drivers.
5. Reboot the computer.
6. Plug in the hardware.

If it is still not recognized, you can try a different computer using a different operating system to rule out operating system issues.

If this does not resolve the issue, contact the vendor of the GPIB controller for assistance.

I'm receiving GPIB timeout errors. What should I do?

If your GPIB controller is recognized by the operating system, but you get a timeout error when you try to communicate with the instrument, check the following:

1. Confirm that the GPIB address you assigned to the instrument is unique and between the range of 0 to 30. It should not be 0 or 21 because they are common controller addresses.
2. Check cabling connection. GPIB cables are heavy and can fall out of the connectors if they are not screwed in securely.
3. Substitute cables to verify cable integrity. For example, if you can send and receive ASCII text, but you cannot do a binary transfer, check your program and the decoding of the binary data. If that does not resolve the problem, try another cable. ASCII text only uses seven data lines in the cable; the binary transfer requires all eight lines.

How do I upgrade the firmware?

⚠ CAUTION

Do not turn off power or remove the USB flash drive until the upgrade process is complete.

NOTE

You can upgrade or downgrade the firmware from the front panel or from the virtual front panel. Refer to Using the Model DMM7510 virtual front panel (on page 2-82) for information.

From the front panel or virtual front panel:

1. Copy the firmware file (.upg file) to a USB flash drive.
2. Verify that the firmware file is in the root subdirectory of the flash drive and that it is the only firmware file in that location.
3. Disconnect any terminals that are attached to the instrument.
4. Turn on instrument power.
5. Insert the flash drive into the USB port on the front panel of the instrument.
6. From the instrument front panel, press the **MENU** key.
7. Under System, select **Info/Manage**.
8. To upgrade to a newer version of firmware, select **Upgrade to New**.
9. To return to a previous version of firmware, select **Downgrade to Older**.
10. If the instrument is controlled remotely, a message is displayed. Select **Yes** to continue.
11. When the upgrade is complete, reboot the instrument.

A message is displayed while the upgrade is in progress.

For additional information about upgrading the firmware, see Upgrading the firmware (on page A-4).

Where can I find updated drivers?

For the latest drivers and additional support information, see the Keithley Instruments support website.

To see what drivers are available for your instrument:

1. Go to the Keithley Instruments support website (*http://www.keithley.com/support*).
2. Enter the model number of your instrument.
3. Select **Software Driver** from the list.

NOTE

If you use the native LabVIEW™ or IVI driver, you must configure the Model DMM7510 to use the SCPI command set. For information on changing the command set, refer to How do I change the command set? (on page 9-7)

How do I change the command set?

You can change the command set that you use with the Model DMM7510. The remote command sets that are available include:

- SCPI: An instrument-specific language built on the SCPI standard.
- TSP: A scripting programming language that contains instrument-specific control commands that can be executed from a stand-alone instrument. You can use TSP to send individual commands or use it to combine commands into scripts.

You cannot combine the command sets.

NOTE

As delivered from Keithley Instruments, the Model DMM7510 is set to work with the Model DMM7510 SCPI command set.

To set the command set from the front panel:

1. Press the **MENU** key.
2. Under System, select **Settings**.
3. Select the button next to Command Set.
4. Select the command set.
5. You are prompted to reboot.

To verify which command set is selected from a remote interface:

Send the command:

```
*LANG?
```

To change to the SCPI command set from a remote interface:

Send the command:

```
*LANG SCPI
```

Reboot the instrument.

To change to the TSP command set from a remote interface:

Send the command:

```
*LANG TSP
```

Reboot the instrument.

Why can't the Model DMM7510 read my USB flash drive?

Verify that the flash drive is formatted with the FAT file system. The Model DMM7510 only supports FAT drives.

In Microsoft® Windows®, you can check the file system by checking the properties of the USB flash drive.

How do I save the present state of the instrument?

You can save the settings in the instrument as a script using the front-panel menus or from a remote interface. After they are saved, you can recall the script or copy it to a USB flash drive.

From the front panel:

1. Configure the Model DMM7510 to the settings that you want to save.
2. Press the **MENU** key.
3. Under Scripts, select **Create Setup**. The CREATE SETUP window is displayed.
4. Select **Create**. A keyboard is displayed.
5. Use the keyboard to enter the name of the script.
6. Select the **OK** button on the displayed keyboard. The script is added to internal memory.

Using SCPI commands:

Configure the instrument to the settings that you want to save. To save the setup, send the command:

```
*SAV <n>
```

Where `<n>` is an integer from 0 to 4.

NOTE
In the front-panel script menus, the setups saved with the `*SAV` command have the name `Setup0x`, where x is the value you set for `<n>`.

Using TSP commands:

Configure the instrument to the settings that you want to save. To save the setup, send the command:

```
createconfigscript("setupName")
```

Where `setupName` is the name of the setup script that will be created.

Why did my settings change?

Many of the commands in the Model DMM7510 are saved with the measure function that was active when you set them. For example, assume you have the measure function set to current and you set a value for display digits. When you change the measure function to voltage, the display digits value changes to the value that was last set for the voltage measure function. When you return to the current measure function, the display digits value returns to the value you set previously.

What is offset compensation?

Offset compensation is a measuring technique that reduces or eliminates thermoelectric EMFs in low-level resistance measurements. The voltage offsets because of the presence of thermoelectric EMFs (V_{EMF}) can adversely affect resistance measurement accuracy.

To overcome these offset voltages, you can use offset-compensated ohms.

What is a configuration list?

A configuration list is a list of stored instrument settings. You can restore these instrument settings to change the active state of the instrument. Configuration lists allow you to record the active state of the instrument, store it, and then return the instrument to that state as needed.

If you are using TSP, configuration lists run faster than a script that is set up to configure the same settings.

The Model DMM7510 supports measure configuration lists, making it possible to sequence through defined measure settings.

Each configuration list consists of a list of configuration points. A configuration point contains all instrument measure settings that were active at a specific point. You can cycle through the configuration point indexes using a trigger model.

For more detail, see Configuration lists (on page 3-32).

Why do I keep seeing the "Undefined header" error?

When you are using the SCPI command language, you may see the −113, "Undefined header," error. This error indicates that what you sent to the instrument did not contain a recognizable command name. The most likely causes for this are:

- A missing space between the command and its parameter. There must be one or more spaces between the command and its parameter. For example,
  `:disp:volt:digits5`
  The correct entry is
  `:disp:volt:digits 5`

- Incorrect short or long form. Check the SCPI command reference (on page 6-1) documentation for the correct command name.

- Spaces in the command name. You cannot use spaces in the command name. For example:
  `syst: err?`
  The correct entry is:
  `:syst:err?`

Why do I see the "Query interrupted" error?

This error occurs when you have sent a valid query to the instrument, then send it another command or query or a Group Execute Trigger (GET) before it has had a chance to send the entire response message (including the line-feed/EOI terminator). The most likely causes are:

- Sending a query to the instrument and then sending another command or query before reading the response to the first query. For example, the following sequence of commands causes an error -410:
  `syst:err?`
  `*opc?`

  You must read the response to `syst:err?` before sending another command or query.

- Incorrectly configured IEEE 488 driver. The driver must be configured so that when talking on the bus it sends line-feed with EOI as the terminator, and when listening on the bus it expects line-feed with EOI as the terminator. See the reference manual for your particular IEEE 488 interface.

Why do I see the "Query unterminated" error?

This error occurs when you address the instrument to talk and it has nothing to say. The most likely causes are:

* A query was not sent. You must send a valid query to the instrument before addressing it to talk. You cannot get a reading until you send the instrument a query.

* An invalid query was sent. If you sent a query and get this error, make sure that the instrument is processing the query without error. For example, sending a query that generates an "Undefined header" error and then addressing the instrument to talk will generate a "Query unterminated" error.

* A valid query in a command string that also contains an invalid command. This can occur when you send multiple commands or queries in one command string (program message). When the instrument detects an error in a command string, it discards all further commands in the command string until the end of the string. For example, this command string would result in a query unterminated error:
`:sens:date?;:sens:func?`

The first command (`:sens:date?`) generates error -113, "Undefined header" and the instrument discards the second command (`:sens:func?`), even though it is a valid query.

Next steps

Additional Model DMM7510 information

For additional information about the Model DMM7510, refer to:

- The Product Information CD-ROM (ships with the product): Contains product documentation
- The Keithley Instruments website (*http://www.keithley.com*) contains the most up-to-date information. From the website, you can access:
    - The Knowledge Center, which contains the following handbooks:
        - *The Low Level Measurements Handbook: Precision DC Current, Voltage, and Resistance Measurements*
        - *Semiconductor Device Test Applications Guide*
    - Application notes
    - Updated drivers
    - Updated firmware
- Your local Field Applications Engineer: They can help you with product selection, configuration, and usage. Check the website for contact information.

Maintenance

In this appendix:

Introduction

The information in this section describes routine maintenance of the instrument that can be performed by the operator.

Line fuse replacement

A fuse located on the Model DMM7510 rear panel protects the power line input of the instrument.

⚠ WARNING

Disconnect the line cord at the rear panel and remove all test leads connected to the instrument before replacing the line fuse. Failure to do so could expose the operator to hazardous voltages that could result in personal injury or death.

Use only the correct fuse type. Failure to do so could result in injury, death, or instrument damage.

Use a 5 x 20 mm slow-blow fuse rated at 250 V, 1 A.

To replace the fuse, you will need a small flat-bladed screwdriver.

Perform the following steps to replace the line fuse:

1. Power off the instrument.
2. Remove all test leads connected to the instrument.
3. Remove the line cord.
4. Locate the fuse drawer, which is above the AC receptacle, as shown in the figure below.

Figure 164: Model DMM7510 line fuse

Line fuse drawer

Tab

5. Use the screwdriver to lift the tab from the fuse drawer.
6. Slide the fuse drawer out. The fuse drawer does not pull completely out of the power module.
7. Snap the fuse out of the drawer.
8. Replace the fuse.
9. Push the fuse drawer back into the module.

If the fuse continues to blow, a circuit malfunction exists and must be corrected. Return the instrument to Keithley Instruments for repair.

Input fuse replacement

The input line from the AMPS connectors on the front and rear panels is protected by two fuses on the rear panel.

Figure 165: DMM7510 rear panel current-input fuse location

⚠ **WARNING**

Make sure the instrument is disconnected from the power line and other equipment before checking or replacing a current-input fuse. Failure to disconnect all power may expose you to hazardous voltages, that, if contacted, could cause personal injury or death. Use appropriate safety precautions when working with hazardous voltages.

⚠ CAUTION
For continued protection against fire or instrument damage, only replace fuse with the type and rating listed. If the instrument repeatedly blows fuses, locate and correct the cause of the problem before replacing the fuse.

To replace a current-input fuse:

1. Turn off the power to the instrument.
2. Disconnect the power line and test leads.
3. From the rear panel, gently push in the AMPS fuse holder and rotate it one-quarter turn counter-clockwise.
4. Remove the fuse, and replace it with the same type (see table below).
5. Install the new fuse by reversing the procedure above.

Manufacturer and part number	Rating	Length
DMM7510-FUSE-3A	3.5 A, 1000 VAC/VDC fast blow	10 mm x 38 mm (0.394" x 1.5")
DMM8510-FUSE-10A	11 A, 1000 VAC/VDC fast blow	10.3 mm x 38 mm (0.406" x 1.5")

NOTE
If the fuse continues to blow, a circuit malfunction exists and must be corrected. Return the instrument to Keithley Instruments for repair.

Front-panel display

Do not use sharp metal objects, such as tweezers or screwdrivers, or pointed objects, such as pens or pencils, to touch the touchscreen. It is strongly recommended that you use only fingers to operate the instrument. Use of clean-room gloves to operate the touchscreen is supported.

Cleaning the front-panel display

If you need to clean the front-panel LCD touchscreen display, use a soft dry cloth.

Abnormal display operation

If the display area is pushed hard during operation, you may see abnormal display operation. To restore normal operation, turn the instrument off and then back on.

Removing ghost images or contrast irregularities

If the display has been operating for a long time with the same display patterns, the display patterns may remain on the screen as ghost images and a slight contrast irregularity may appear. Note that if this occurs, it does not adversely affect the performance reliability of the display.

To regain normal operation, stop using the front-panel display for some time. You can turn off the front-panel display while continuing operation using remote commands and the virtual front panel.

To turn off the front-panel display using a SCPI command:

Send the command:

```
DISPlay:LIGHt:STATe OFF
```

To turn off the front-panel display using a TSP command:

Send the command:

```
display.lightstate = display.STATE_LCD_OFF
```

Upgrading the firmware

To upgrade the Model DMM7510 firmware, you load an upgrade file into the instrument. You can load the file from the front-panel USB port using either a remote interface or the front panel of the instrument. If you are using Test Script Builder (TSB), you can upgrade the firmware from TSB using a file saved to the computer on which TSB is running.

During the upgrade to new process, the instrument verifies that the version you are loading is newer than what is on the instrument. If the version is older or at the same revision level, no changes are made.

If you want to return to a previous version or reload the present version of the firmware, select **Downgrade to Older**. This forces the instrument to load the firmware regardless of the version.

The upgrade process normally takes about five minutes.

Upgrade files are available on the [Keithley Instruments website](http://www.keithley.com) (*http://www.keithley.com*).

To find firmware files on the Keithley Instruments website:
1. Select the **Support** tab.
2. In the model number box, type **DMM7510**.
3. Select **Firmware**.
4. Click the search button. A list of available firmware updates and any available documentation for the instrument is displayed.
5. Click the file you want to download.

⚠ CAUTION

Disconnect the input terminals before you upgrade or downgrade.

Do not remove power from the Model DMM7510 or remove the USB flash drive while an upgrade or downgrade is in progress. Wait until the instrument completes the procedure and shows the opening display. If you are upgrading a Model DMM7510-NFP instrument, the LAN and 1588 LEDs on the front panel blink during the upgrade and stop when the upgrade is complete.

From the front panel

> ⚠ CAUTION
>
> Do not turn off power or remove the USB flash drive until the upgrade process is complete.

> NOTE
>
> You can upgrade or downgrade the firmware from the front panel or from the virtual front panel. Refer to Using the Model DMM7510 virtual front panel (on page 2-82) for information.

From the front panel or virtual front panel:

1. Copy the firmware file (`.upg` file) to a USB flash drive.
2. Verify that the firmware file is in the root subdirectory of the flash drive and that it is the only firmware file in that location.
3. Disconnect any terminals that are attached to the instrument.
4. Turn on instrument power.
5. Insert the flash drive into the USB port on the front panel of the instrument.
6. From the instrument front panel, press the **MENU** key.
7. Under System, select **Info/Manage**.
8. To upgrade to a newer version of firmware, select **Upgrade to New**.
9. To return to a previous version of firmware, select **Downgrade to Older**.
10. If the instrument is controlled remotely, a message is displayed. Select **Yes** to continue.
11. When the upgrade is complete, reboot the instrument.

A message is displayed while the upgrade is in progress.

Using TSP

> ⚠ CAUTION
>
> Do not turn off power or remove the USB flash drive until the upgrade process is complete.

Using TSP over a remote interface:

1. Copy the firmware upgrade file to a USB flash drive.
2. Verify that the upgrade file is in the root subdirectory of the flash drive and that it is the only firmware file in that location.
3. Disconnect the input and output terminals that are attached to the instrument.
4. Power on the instrument.
5. Insert the flash drive into the USB port on the front panel of the instrument.
6. To upgrade to a newer version of firmware, send:
   ```
   upgrade.unit()
   ```
7. To return to a previous version of firmware, send:
   ```
   upgrade.previous()
   ```
8. After completion of the upgrade, reboot the instrument.

A message is displayed on the front panel of the instrument while the upgrade is in progress. In addition, the LEDs in the upper right of the front panel blink while the upgrade is in process.

Using SCPI

There are no SCPI commands that you can use to upgrade the firmware. To upgrade the firmware, you must either use the front panel, virtual front panel, or switch the command set to TSP.

To use the front panel to upgrade the firmware, see .

⚠ CAUTION
Do not turn off power or remove the USB flash drive until the upgrade process is complete.

If you need to upgrade the firmware from a remote interface and you are using a SCPI command set:

1. Copy the firmware upgrade file to a USB flash drive.
2. Verify that the upgrade file is in the root subdirectory of the flash drive and that it is the only firmware file in that location.
3. Disconnect the input and output terminals that are attached to the instrument.
4. Power on the instrument.
5. Change the command set to TSP by sending the command:
    `*LANG TSP`
6. Turn the instrument off and then turn it on again.
7. Insert the flash drive into the USB port on the front panel of the instrument.
8. To upgrade to a newer version of firmware, send:
    `upgrade.unit()`
9. To return to a previous version of firmware, send:
    `upgrade.previous()`
10. After completion of the upgrade, turn the instrument off and then turn it on again.
11. To return to the SCPI command set, send the command:
    `*LANG SCPI`
12. Turn the instrument off and then turn it on again.

A message is displayed on the front panel of the instrument while the upgrade is in process. In addition, the LEDs in the upper right of the front panel blink while the upgrade is in process.

Using TSB

⚠ CAUTION

Do not turn off power or remove the USB flash drive until the upgrade process is complete.

You can use Test Script Builder (TSB) to upgrade the firmware of your instrument.

To upgrade the firmware using Test Script Builder:

1. Disconnect the input and output terminals that are attached to the instrument.
2. Start Test Script Builder.
3. On the Instrument Console toolbar, click the **Open Instrument** icon.

Figure 166: TSB Instrument Console toolbar

4. Select your communication interface from the Select Instrument dialog box. See the Connecting an instrument in TSB (on page 7-33) for details on opening communications.
5. On the Instrument Console toolbar, click the View Menu icon. Select **Instrument**, then select **Flash**.
6. From the Select a Firmware Image File dialog box, use the browser to select the file name of the new firmware or enter the path and file name.
7. If you are upgrading the firmware, replace the existing firmware with a newer version of firmware.
8. If you are downgrading the firmware, replace the existing firmware with an older version of firmware or repair the same version.
9. Click **OK**. A Progress Information bar is displayed on the instrument during the update. In addition, the LEDs in the upper right of the front panel blink while the upgrade is in process.
10. Wait until the instrument indicates that the firmware upgrade is complete. (TSB may indicate that the upgrade is complete before it is finalized on the instrument.)
11. Reboot the instrument.

Common commands

In this appendix:

Introduction

This section describes the general remote interface commands and common commands. Note that although these commands are essentially the same as those defined by the IEEE Std 488.2 standard, the Model DMM7510 does not strictly conform to that standard.

The general remote interface commands are commands that have the same general meaning, regardless of the instrument you use them with (for example, DCL always clears the GPIB interface and returns it to a known state).

The common commands perform operations such as reset, wait-to-continue, and status.

Common commands always begin with an asterisk (*) and may include one or more parameters. The command keyword is separated from the first parameter by a blank space.

If you are using the TSP remote interface, each command must be sent in a separate message.

If you are using a SCPI remote interface, the commands can be combined. Use a semicolon (;) to separate multiple commands, as shown below:

```
*RST; *CLS; *ESE 32; *OPC?
```

Although the commands in this section are shown in uppercase, they are not case sensitive (you can use either uppercase or lowercase).

NOTE

If you are using the TSP remote interface, note that the common commands and general bus commands cannot be used in scripts.

*CLS

This command clears the event registers and queues.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
*CLS
```

Details

This command clears the event registers of the Questionable Event and Operation Event Register set. It also clears the event log. It does not affect the Questionable Event Enable or Operation Event Enable registers.

This is the equivalent of sending the SCPI commands `:STATus:CLEar` and `:SYStem:CLEar` or the TSP commands `status.clear()` and `eventlog.clear()`.

To reset all the bits of the Standard Event Enable Register, send the command:

```
*ESE 0
```

Also see

[*ESE](#) (on page B-2)
[:STATus:PRESet](#) (on page 6-126)

*ESE

This command sets and queries bits in the Status Enable register of the Standard Event Register.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Not applicable	See **Details**

Usage

```
*ESE <n>
*ESE?
```

`<n>`	The value of the Status Enable register of the Standard Event Register (0 to 255)

Details

When a bit in the Status Enable register is set on and the corresponding bit in the Standard Event Status register is set on, the ESB bit of the Status Byte Register is set to on.

To set a bit on, send the constant or the value of the bit as the `<n>` parameter.

If you are using TSP, you can set the bit as a constant or a numeric value, as shown in the table below. To set more than one bit of the register, you can send multiple constants with + between them. You can also set *standardRegister* to the sum of their decimal weights. For example, to set bits B0 and B2, set *standardRegister* to 5 (which is the sum of 1 + 4). You can also send:

```
status.standard.enable = status.standard.OPC + status.standard.QYE
```

If you are using SCPI, you can only set the bit as a numeric value.

When zero (0) is returned, no bits are set. You can also send 0 to clear all bits.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Bit	Decimal value	Constant	When set, indicates the following has occurred:
0	1	`status.standard.OPC`	All pending selected instrument operations are complete and the instrument is ready to accept new commands. The bit is set in response to an [*OPC](#) (on page B-6) command or TSP [opc()](#) (on page 8-228) function.
1	2	Not used	Not used.
2	4	`status.standard.QYE`	Attempt to read data from an empty Output Queue.
3	8	Not used	Not used.
4	16	Not used	Not used.
5	32	Not used	Not used.
6	64	Not used	Not used.
7	128	`status.standard.PON`	The instrument has been turned off and turned back on since the last time this register was read.

Command errors include:

- **IEEE Std 488.2 syntax error:** The instrument received a message that does not follow the defined syntax of the IEEE Std 488.2 standard.
- **Semantic error:** The instrument received a command that was misspelled or received an optional IEEE Std 488.2 command that is not implemented in the instrument.
- **GET error:** The instrument received a Group Execute Trigger (GET) inside a program message.

NOTE

Constants are only available if you are using the TSP command set. If you are using the SCPI command set, you must use the decimal values.

Example

`*ESE 129`	`*ESE 129` sets the Status Enable register of the Standard Event Register to binary `10000001`, which enables the PON and OPC bits.

Also see

[*CLS](#) (on page B-2)
[Standard Event Register](#) (on page C-3)
[Status model](#) (on page C-1)

*ESR?

This command reads and clears the contents of the Standard Event Status Register.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
*ESR?
```

Details

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Bit	Decimal value	Constant	When set, indicates the following has occurred:
0	1	`status.standard.OPC`	All pending selected instrument operations are complete and the instrument is ready to accept new commands. The bit is set in response to an *OPC (on page B-6) command or TSP opc() (on page 8-228) function.
1	2	Not used	Not used.
2	4	`status.standard.QYE`	Attempt to read data from an empty Output Queue.
3	8	Not used	Not used.
4	16	Not used	Not used.
5	32	Not used	Not used.
6	64	Not used	Not used.
7	128	`status.standard.PON`	The instrument has been turned off and turned back on since the last time this register was read.

Command errors include:

- **IEEE Std 488.2 syntax error:** The instrument received a message that does not follow the defined syntax of the IEEE Std 488.2 standard.
- **Semantic error:** The instrument received a command that was misspelled or received an optional IEEE Std 488.2 command that is not implemented in the instrument.
- **GET error:** The instrument received a Group Execute Trigger (GET) inside a program message.

Example

`*ESR?`	Example output: `128` Shows that the Standard Event Status Register contains binary `10000000`, which indicates that the instrument was rebooted since the last time this register was read.

Also see

Status model (on page C-1)

*IDN?

This command retrieves the identification string of the instrument.

Type	Affected by	Where saved	Default value
Query only	None	Not applicable	Not applicable

Usage

```
*IDN?
```

Details

The identification string includes the manufacturer, model number, serial number, and firmware revision of the instrument. The string is formatted as follows:

```
KEITHLEY INSTRUMENTS,MODEL nnnn,xxxxxxxx,yyyyyy
```

Where:

- `nnnn` is the model number
- `xxxxxxxx` is the serial number
- `yyyyyy` is the firmware revision level

Example

`*IDN?`	Output: `KEITHLEY INSTRUMENTS,MODEL DMM7510,01234567,1.0.0i`

Also see

System information (on page 2-86)

*LANG

This command determines which command set is used by the instrument.

Type	Affected by	Where saved	Default value
Command and query	Not applicable	Nonvolatile memory	SCPI

Usage

```
*LANG <commandSet>
*LANG?
```

`<commandSet>`	The command set to be used: • `TSP` • `SCPI`

Details

The remote command sets that are available include:

- SCPI: An instrument-specific language built on the SCPI standard.
- TSP: A scripting programming language that contains instrument-specific control commands that can be executed from a stand-alone instrument. You can use TSP to send individual commands or use it to combine commands into scripts.

You cannot combine the command sets.

Example

`*LANG TSP` `*LANG?`	Set the command set to TSP. Verify setting by sending the command set query. Output: `TSP` The TSP command set is in use.

Also see

Status model (on page C-1)

*OPC

This command sets the operation complete (OPC) bit after all pending commands, including overlapped commands, have been executed.

Type	Affected by	Where saved	Default value
Command and query			

Usage

```
*OPC
*OPC?
```

Details

When `*OPC` is sent, the OPC bit (bit 0) in the Status Event Status Register is set after all pending command operations have been executed. After all programmed operations are complete, the instrument returns to idle, at which time all pending commands (including `*OPC` and `*OPC?`) are executed. After the last pending command is executed, the OPC bit is set or an ASCII "1" is placed in the Output Queue.

When the trigger model is executing, most sent commands are not executed. If a command cannot be processed, an error event message is generated in the event log.

Also see

:INITiate[:IMMediate] (on page 6-166)
opc() (on page 8-228)

*RST

This command resets the instrument settings to their default values and clears the reading buffers.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
*RST
```

Details

Returns the instrument to default settings, cancels all pending commands, and cancels the response to any previously received `*OPC` and `*OPC?` commands.

Also see

reset() (on page 8-233)

*SRE

This command sets or clears the bits of the Service Request Enable Register.

Type	Affected by	Where saved	Default value
Command and query	:STATus:PRESet status.preset()	Not applicable	0

Usage

```
*SRE <n>
*SRE?
```

<n>	Clear the Status Request Enable Register: 0 Set the instrument for an SRQ interrupt: 32

Details

This command sets or clears the individual bits of the Status Request Enable Register.

The Status Request Enable Register is cleared when power is cycled or when a parameter value of 0 is sent with this command.

The instrument returns a decimal value that corresponds to the binary-weighted sum of all bits set in the register.

Bit	Decimal value	Constants	When set, indicates the following has occurred:
0	1	status.MSB	An enabled event in the Measurement Event Register has occurred.
1	2	Not used	Not used.
2	4	status.EAV	An error or status message is present in the Error Queue.
3	8	status.QSB	An enabled event in the Questionable Status Register has occurred.
4	16	status.MAV	A response message is present in the Output Queue.
5	32	status.ESB	An enabled event in the Standard Event Status Register has occurred.
6	64	Not used	Not used.
7	128	status.OSB	An enabled event in the Operation Status Register has occurred.

NOTE

Constants are only available if you are using the TSP command set. If you are using the SCPI command set, you must use the decimal values.

Example

*SRE 0	Clear the bits of the Status Request Enable Register.

Also see

Understanding bit settings (on page C-14)

*STB?

This command gets the status byte of the instrument without clearing the request service bit.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	Not applicable

Usage

```
*STB?
```

Details

This command is similar to a serial poll, but it is processed like any other instrument command.

The *STB? command returns the same result as a serial poll, but the master summary bit (MSB) is not cleared if a serial poll has occurred. The MSB is not cleared until all other bits feeding into the MSB are cleared.

Example

*STB?	Queries the status byte.

Also see

None

*TRG

This command generates a trigger event from a remote command interface.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
*TRG
```

Details

Use the *TRG command to generate a trigger event.

If you are using the SCPI command set, this command generates the COMMand event. If you are using the TSP command set, this command generates the trigger.EVENT_COMMAND event. You can use this constant as the stimulus of any trigger object, which causes that trigger object to respond to the trigger events generated by *TRG. See Using trigger events to start actions in the trigger model (on page 3-97).

Also see

:INITiate[:IMMediate] (on page 6-166)

*TST?

This command is accepted and returns 0. A self-test is not actually performed.

Type	Affected by	Where saved	Default value
Query only	Not applicable	Not applicable	0

Usage

```
*TST?
```

Also see

None

*WAI

This command postpones the execution of subsequent commands until all previous overlapped commands are finished.

Type	Affected by	Where saved	Default value
Command only	Not applicable	Not applicable	Not applicable

Usage

```
*WAI
```

Details

There are two types of instrument commands:

- **Overlapped commands:** Commands that allow the execution of subsequent commands while instrument operations of the overlapped command are still in progress.
- **Sequential commands:** Commands whose operations must finish before the next command is executed.

The *WAI command suspends the execution of commands until the instrument operations of all previous overlapped commands are finished. The *WAI command is not needed for sequential commands. Typically, this command is sent after the initiate trigger model command.

Also see

Status model

In this appendix:

Overview

The status model consists of status register sets and queues. You can monitor the status model to view instrument events and configure the status model to control the events.

As you work with the status model, be aware that the end result applies to the Status Byte Register. All the status register sets and queues flow into the Status Byte Register. Your test program can read this register to determine if a service request (SRQ) has occurred, and if so, which event caused it.

The Status Byte Register, register sets, and queues include:

- Standard Event Register
- Questionable Event Register
- Operation Event Register
- Output Queue
- Error Queue

The relationship between the Status Byte Register, Standard Event Register, event queue, and output queue is shown in the . The relationship between the Status Byte Register, Questionable Event Register, and the Operation Event Register is shown in the .

Figure 167: Non-programmable status registers diagram

Standard Event Register

The Standard Event Register set includes two 8-bit registers:

- **Standard Event Status Register:** Reports when a predefined event has occurred. The register latches the event and the corresponding bit remains set until it is cleared by a read.

- **Standard Event Status Enable Register:** You can enable or disable bits in this register. This allows the predefined event (from the Standard Event Status Register) to set the ESB of the Status Byte Register.

Figure 168: Model DMM7510 Standard Event Register

PON = Power On
QYE = Query Error
OPC = Operation Complete
& = Logical AND
OR = Logical OR

Bit	When set, indicates the following has occurred:
0	Operation complete: All pending selected instrument operations are complete and the instrument is ready to accept new commands. The bit is set in response to an *OPC (on page B-6) command or TSP opc() (on page 8-228) function.
1	Not used.
2	Query error: Attempt to read data from an empty Output Queue.
3	Not used.
4	Not used.
5	Not used.
6	Not used.
7	Power-on: The instrument has been turned off and turned back on since the last time this register was read.

You can use the following commands to read and set bits contained in the Standard Event Register.

Description	SCPI command	TSP command
Read the Standard Event Status Register	*ESR? (on page B-4)	status.standard.event (on page 8-249)
Set or read the OR bits in the Standard Event Status Enable Register	*ESE (on page B-2) ESE?	status.standard.enable (on page 8-248)

Programmable status register sets

You can program the registers in the Questionable Event Register and Operation Event Register sets.

These event registers contain bits that identify the state of an instrument condition or event. They also contain bits that determine if those events are sent to the Status Byte Register. You can enable the events, which causes the associated bit to be set in the Status Byte Register.

The Questionable and Operation Event Registers are identical except that they set different bits in the Status Byte Register. The Questionable Event Registers set the MSB and QSM bits. The Operation Event Registers set the OSB bit.

Each 16-bit register set includes the following registers:

- **Condition:** A read-only register that is constantly updated to reflect the present operating conditions of the instrument. You can determine which events set or clear the bits.

- **Event:** A read-only register that sets a bit to 1 when an applicable event occurs. The bit remains at 1 until the register is reset. This register is reset when power is cycled, when a `*CLS` command is sent, or when the register is read. You can determine which events set the bits.

- **Event enable:** A read-write register that determines which events set the summary bit in the Status Byte Register. For example, if a bit is a 1 in the event register and the corresponding bit is a 1 in the Event Enable Register, bits in the Status Byte Register are set. If the event enable bit is set in the Questionable Event Registers, the event sets the MSB and QSM bits in the Status Byte Register. If the event enable bit is set in the Operation Event Registers, the event sets the OSB bit in the Status Byte Register.

When the instrument is powered on, all bits in the Questionable Event and Operation Event Registers are set to 0.

Figure 169: Programmable status registers diagram

Questionable Event Register

You can program the bits in the Questionable Event Register to be cleared or set when an event occurs.

When an enabled Questionable Event Register bit is set (because the enabled event occurs), the corresponding bit B0 (MSB) and Bit B3 (QSB) of the Status Byte Register is set. The corresponding Questionable Event Register Condition Register reflects the present status of the instrument, so it is set while the event occurs.

When reading a register, a numeric value is returned. The binary equivalent of this value indicates which bits in the register are set. For details, see Understanding bit settings (on page C-14).

You can use the following commands to read and set bits contained in the Questionable Event Register.

Description	SCPI command	TSP command
Read the Questionable Condition Register	:STATus:QUEStionable:CONDition? (on page 6-127)	status.questionable.condition (on page 8-243)
Set or read the contents of the Questionable Event Enable Register	:STATus:QUEStionable:ENABle (on page 6-127)	status.questionable.enable (on page 8-244)
Read the Questionable Event Register	:STATus:QUEStionable[:EVENt]? (on page 6-129)	status.questionable.event (on page 8-244)
Request the mapped set event and mapped clear event status for a bit in the Questionable Event Register	:STATus:QUEStionable:MAP (on page 6-128)	status.questionable.getmap() (on page 8-245)
Map event to a bit in the Questionable Event Register	:STATus:QUEStionable:MAP (on page 6-128)	status.questionable.setmap() (on page 8-246)

Operation Event Register

You can program the bits in the Operation Condition and Operation Event Status Registers to be cleared or set when an event occurs.

When an enabled Operation Event Register bit is set (because the enabled event occurs), the corresponding bit B7 (OSB) of the Status Byte Register is set. The corresponding Operation Event Register Condition Register reflects the present status of the instrument, so it will be set while the event occurs.

You can use the following commands to read and set bits contained in the Operation Event Register.

Description	SCPI command	TSP command
Read the Operation Condition Register	:STATus:OPERation:CONDition? (on page 6-123)	status.operation.condition (on page 8-239)
Set or read the contents of the Operation Event Enable Register	:STATus:OPERation:ENABle (on page 6-124)	status.operation.enable (on page 8-239)
Read the Operation Event Register	:STATus:OPERation[:EVENt]? (on page 6-126)	status.operation.event (on page 8-240)
Request the mapped set event and mapped clear event status for a bit in the Operation Event Registers	:STATus:OPERation:MAP (on page 6-125)	status.operation.getmap() (on page 8-241)
Map events to bit in the Operation Event Register	:STATus:OPERation:MAP (on page 6-125)	status.operation.setmap() (on page 8-242)

Mapping events to bits

To program the Questionable and Operation Event Registers, you map events to specific bits in the register. This causes a bit in the condition and event registers to be set (or cleared) when the specified event occurs. You can map events to bits B0 through B14 (bit B15 is always set to zero).

When you have a mapped-set event, the bits in the corresponding condition register and event register are set when the mapped-set event is detected. The bits remain at 1 until the event register is read or the status model is reset.

When you have a mapped-clear event, the bit in the condition register is cleared to 0 when the event is detected.

You can map any event to any bit in these registers. An event is the number that accompanies an error, warning, or informational message that is reported in the event log. For example, for the error "Error –221, Settings Conflict," the event is –221. Note that some informational messages do not have a related event number, so they cannot be mapped to a register.

You do not need to map clear events to generate SRQs. However, if you want to read the condition register to report status, you must map both a set event and a clear event. If no clear event is mapped, the bits are cleared only when the instrument power is turned off and turned on.

You can use the following SCPI commands to read and map events to bits in the programmable registers:

- :STATus:OPERation:MAP (on page 6-125)
  This command maps the set and clear events to a specified operation event register bit. Use the query form of this command to read the mapped set and clear status.

- :STATus:QUEStionable:MAP (on page 6-128)
  This command maps the set and clear events to a specified operation event register bit. Use the query form of this command to read the mapped set and clear status.

You can use the following TSP commands to read and map events to bits in the programmable registers:

- status.operation.getmap() (on page 8-241)
  This command reads the mapped set and clear status for the specified operation event bit.

- status.operation.setmap() (on page 8-242)
  This command maps the set and clear events to a specified operation event register bit.

- status.questionable.getmap() (on page 8-245)
  This command reads the mapped set and clear status for the specified questionable event bit.

- status.questionable.setmap() (on page 8-246)
  This command maps the set and clear events to a specified questionable event register bit.

You can map any event that appears with a number in the event queue to any available bit in a programmable register. The programmable registers and their relationships to the Status Byte Register are shown in the Programmable status registers diagram (on page C-6). The following example event queue log entries contain actual events that can be mapped to a status model bit.

```
2731 Trigger Model Initiated "Trigger model #1 has been initiated"
2732 Trigger Model Idle  "Trigger model #1 has been idled"
4916 Reading buffer cleared "Reading buffer <buffer name> is  0% filled"
4917 Reading buffer full "Reading buffer <buffer name> is  100% filled"
```

See Using the event log (on page 2-152) for additional information on finding events.

Status Byte Register

The Status Byte Register monitors the registers and queues in the status model and generates service requests (SRQs).

When bits are set in the status model registers and queues, they generate summary messages that set or clear bits of the Status Byte Register. You can enable these bits to generate an SRQ.

Service requests (SRQs) instruct the controller that the instrument needs attention or that some event has occurred. When the controller receives an SRQ, the controller can interrupt existing tasks to perform tasks that address the request for service.

For example, you might program your instrument to send an SRQ when a specific instrument error event occurs. To do this, you set the Status Request Enable bit 2 (EAV). In this example, the following actions occur:

* The error event occurs.
* The error event is logged in the Error Queue.
* The Error Queue sets the EAV bit of the Status Byte Register.
* The EAV bits are summed.
* The RQS bit of the Status Byte Register is set.
* On a GPIB system, the SRQ line is asserted. On a VXI-11 or USB system, an SRQ event is generated.

For an example of this, see the example code provided in SRQ on error (on page C-16).

The summary messages from the status registers and queues set or clear the appropriate bits (B0, B2, B3, B4, B5, and B7) of the Status Byte Register. These summary bits do not latch, and their states (0 or 1) are solely dependent on the summary messages (0 or 1). For example, if the Standard Event Register is read, its register will clear. As a result, its summary message resets to 0, which in turn resets the ESB bit in the Status Byte Register.

The Status Byte Register also receives summary bits from itself, which sets the Master Summary Status (MSS) bit.

When using the GPIB, USB, or VXI-11 serial poll sequence of the Model DMM7510 to get the status byte (serial poll byte), bit B6 is the RQS bit. See Serial polling and SRQ (on page C-12) for details on using the serial poll sequence.

When using the `*STB?` common command or `status.condition` command to read the status byte, bit B6 is the MSS bit.

To reset the bits of the Service Request Enable Register to 0, use 0 as the parameter value for the command (for example, `*SRE 0` or `status.request_enable = 0`).

You can read and set which bits to AND in the Status Byte Register using the following commands.

Description	SCPI command	TSP command
Read the Status Byte Register	*STB? (on page B-9)	status.condition (on page 8-238)
Read the Status Request Enable Register	*SRE (on page B-8)	status.request_enable (on page 8-246)
Enable bits in the Status Request Enable Register	*SRE (on page B-8)	status.request_enable (on page 8-246)

Status Byte Register diagram

The Status Byte Register consists of two 8-bit registers that control service requests, the Status Byte Register and the Service Request Enable Register. These registers are shown in the following figure.

Figure 170: Model DMM7510 Status Byte Register

The bits in the Status Byte Register are described in the following table.

Bit	Decimal value	Bit name	When set, indicates the following has occurred:
0	1	Measurement summary Bit (MSB)	An enabled questionable event
1	2	Not used	Not applicable
2	4	Error available (EAV)	An error is present in the error queue (warning and information messages do not affect this bit)
3	8	Questionable summary bit (QSB)	An enabled questionable event
4	16	Message available (MAV)	A response message is present in the output queue
5	32	Event summary bit (ESB)	An enabled standard event
6	64	Request for service (RQS)/Master summary status (MSS)	An enabled summary bit of the Status Byte Register is set; depending on how it is used, this is either the Request for Service (RQS) bit or the Master Summary Status (MSS) bit
7	128	Operation summary bit (OSB)	An enabled operation event

Service Request Enable Register

This register is programmed by the user and is used to enable or disable the setting of bit B6 (RQS/MSS) by the Status Summary Message bits (B0, B1, B2, B3, B4, B5, and B7) of the Status Byte Register. As shown in the Status Byte Register (on page C-9) topic, a logical AND operation is performed on the summary bits (&) with the corresponding enable bits of the Service Request Enable Register. When a logical AND operation is performed with a set summary bit (1) and with an enabled bit (1) of the enable register, the logic "1" output is applied to the input of the logical OR gate and, therefore, sets the MSS/RQS bit in the Status Byte Register.

You can set or clear the individual bits of the Service Request Enable Register by using the `*SRE` common command or `status.request_enable`. To read the Service Request Enable Register, use the `*SRE?` query or `print(status.request_enable)`. The Service Request Enable Register clears when power is cycled or a parameter value of 0 is sent with a status request enable command (for example, a `*SRE 0` or `status.request_enable = 0` is sent). You can program and read the SRQ Enable Register using the following commands.

Description	SCPI command	TSP command
Read the Status Request Enable Register	*SRE (on page B-8)	status.request_enable (on page 8-246)
Enable bits in the Status Request Enable Register	*SRE (on page B-8)	status.request_enable (on page 8-246)

Queues

The instrument includes an Output Queue and an Error Queue. The Output Queue holds messages from readings and responses. The Error Queue holds error event messages from the event log. Both are first-in, first-out (FIFO) registers.

Output Queue

The output queue holds response messages to SCPI query and TSP `print()` commands.

When data is placed in the Output Queue, the Message Available (MAV) bit in the Status Byte Register is set. The bit is cleared when the Output Queue is empty.

To clear data from the Output Queue, read the messages. To read a message from the Output Queue, address the instrument to talk after the appropriate query is sent.

Error Queue

The Error Queue holds any error events that are posted in the event log. When an error event occurs, it is posted to the Error Queue, which sets the Error Available (EAV) bit in the Status Byte Register.

The instrument clears error event messages from the event log when it retrieves the event log. When the error event messages are cleared from the event log, the EAV bit in the Status Byte Register is cleared.

You can clear the Error Queue by sending the common command `*CLS` or the TSP command `status.clear()`. Note that `status.clear()` also clears all event registers.

For information regarding the event log, see Using the event log (on page 2-152).

Serial polling and SRQ

Any enabled event summary bit that goes from 0 to 1 sets bit B6 and generates a service request (SRQ).

In your test program, you can periodically read the Status Byte to check if an SRQ has occurred and what caused it. If an SRQ occurs, the program can, for example, branch to an appropriate subroutine that will service the request.

SRQs can be managed by the serial poll sequence of the instrument. If an SRQ does not occur, bit B6 (RQS) of the Status Byte Register remains cleared, and the program proceeds normally after the serial poll is performed. If an SRQ does occur, bit B6 of the Status Byte Register is set, and the program can branch to a service subroutine when the SRQ is detected by the serial poll.

The serial poll automatically resets RQS of the Status Byte Register. This allows subsequent serial polls to monitor bit B6 for an SRQ occurrence that is generated by other event types.

For common commands and TSP commands, B6 is the MSS (Message Summary Status) bit. The serial poll does not clear the MSS bit. The MSS bit remains set until all enabled Status Byte Register summary bits are reset.

Programming enable registers

You can program the bits in the enable registers of the Status Model registers.

When you program an enable register bit to 0, no action occurs if the bits in the corresponding registers are set (1).

When you program an enable register bit to 1, if the bits in the corresponding registers are set (1), the AND condition occurs and a bit in the Status Byte Register is set to (1).

You must program all bits in an enable register at the same time. This means you need to determine what each bit value in the register will be, then add them together to determine the value of all the bits in the register. See Understanding bit settings (on page C-14) for more information on determining the value of the bits in the registers.

For example, you might want to enable the Standard Event Register to set the ESB bit in the Status Byte Register whenever an operation complete occurs or whenever an operation did not execute properly because of an internal condition. To do this, you need to set bits 0 and 3 of the Standard Event Register to 1. These bits have decimal values of 1 and 8, so to set both bits to 1, you set the register to 9.

Using SCPI:

Send the command:

```
*ese 9
```

Using TSP

Send the command:

```
status.standard.enable = 9
```

Reading the registers

You can read any register in the status model. The response is a decimal value that indicates which bits in the register are set. See <u>Understanding bit settings</u> (on page C-14) for information on how to convert the decimal value to bits.

Using SCPI commands:

If you are using SCPI, you use the query commands in the STATus subsystem and common commands to read registers.

Using TSP commands:

If you are using TSP, you print the TSP command to read the register. You can use either `print()`, which returns the decimal value, or `print(tostring())`, which returns the string equivalent of the decimal value.

You can also send the common commands to read the register.

For example, you can send any one of the following commands to read the Status Enable Register of the Standard Event Register:

```
print(status.standard.enable)
*ese?
print(tostring(status.standard.enable))
```

Understanding bit settings

When you write to or read a status register, you can use binary, decimal, or hexadecimal values to represent the binary values of the bit states. When the value is converted to its binary equivalent, you can determine which bits are set on or clear. Zero (0) indicates that all bits are clear.

In the Model DMM7510, the least significant bit is always bit B0. The most significant bit differs for each register, but in most cases is either bit B7 or bit B15.

Bit position	B7	B6	B5	B4	B3	B2	B1	B0
Binary value	1000 0000	0100 0000	0010 0000	0001 0000	1000	0100	0010	0000
Decimal value	128	64	32	16	8	4	2	1
Weight	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Bit position	B15	B14	B13	B12	B11	B10	B9	B8
Binary value	1000 0000 0000 0000	0100 0000 0000 0000	0010 0000 0000 0000	0001 0000 0000 0000	1000 0000 0000	0100 0000 0000	0010 0000 0000	0001 0000 0000
Decimal value	32768	16384	8192	4096	2048	1024	512	256
Weight	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8

For example, if a value of `1.29000e+02` (which is 129) is read as the value of the condition register, the binary equivalent is 0000 0000 1000 0001. This value indicates that bit B0 and bit B7 are set and all other bits are cleared.

For example, if you read a value of 1.22880e+04 (12,288) for the condition register, the binary equivalent is 0011 0000 0000 0000. This value indicates that bits B12 and B13 are set.

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

When bit B12 (4096) and bit B13 (8192) are set (1), the decimal equivalent is 4096 + 8192 = 12,288.

Clearing registers

Registers in the status model can be cleared using commands or by instrument actions. When a register is cleared, the bits in the register are set to 0.

The event log and all registers are cleared when instrument power is cycled.

When you read a bit from the Operation Event, Questionable Event, or Standard Event Status Register, the entire 16-bit or 8-bit register value is returned. The event register is cleared or set to 0.

Using SCPI commands:

To clear the event registers of the Questionable Event Status Register and Operation Event Status Register sets, Standard Event Register, and Status Byte Register, send:

```
*CLS
```

When using the SCPI interface, this command does not affect the Questionable Event Enable Register and Operation Event Enable Register sets.

To clear the Questionable Event Status Register, the Operation Event Status Register sets, Standard Event Register, and Status Byte Register, send:

```
STATus:CLEar
```

When using the SCPI interface, this command does not affect the Questionable Event Enable Register or Operation Event Enable Register sets.

Using TSP commands:

To clear the event registers of the Questionable Event Status Register and Operation Event Status Register sets, Standard Event Register, and Status Byte Register send:

```
*CLS
```

To clear the Questionable Event Status Register, the Operation Event Status Register sets, Standard Event Register, and Status Byte Register send:

```
status.clear()
```

When using the SCPI interface, this command does not affect the Questionable Event Enable Register or Operation Event Enable Register sets.

Status model programming examples

The following examples illustrate how to generate an SRQ using the status model.

SRQ on error

This example shows you how to generate an SRQ when an instrument error event occurs.

Using SCPI commands:

```
*RST
SYST:CLE
STAT:CLE
*SRE 4
MAKEERROR
```

Using TSP commands:

```
reset()
-- Clear Error Queue so EAV bit can go low.
errorqueue.clear()

-- Clear the status byte.
status.clear()

-- Enable SRQ on error available.
status.request_enable = status.EAV

-- Send a line of code that will generate an error event.
beeper = 1
```

SRQ when reading buffer becomes full

This example shows you how to generate an SRQ when the Model DMM7510 reading buffer is full. You can use this to notify the controlling computer that it needs to read back the data and empty the buffer. After configuring the status model, this code configures the default reading buffer 1 to a size of 100, and then configures the Model DMM7510 to fill the buffer. After the buffer is full, the instrument generates an SRQ and returns the data.

Using SCPI commands:

```
*RST
STAT:CLE
STAT:OPER:MAP 0, 4917, 4916
STAT:OPER:ENAB 1
*SRE 128
TRAC:CLE
TRAC:POIN 100, "defbuffer1"
COUNT 100
READ? "defbuffer1"
TRAC:DATA? 1, 100, "defbuffer1", READ
```

Using TSP commands:

```
reset()
-- Clear the status byte
status.clear()

-- Map bit 0 of operational status register to set on buffer
-- full (4917) and clear on buffer empty (4916).
status.operation.setmap(0, 4917, 4916)

-- Enable bit 0 to flow through to the status byte.
status.operation.enable = 1

-- Enable the Operational Summary Bit to set the Master
-- Summary Bit/RQS
status.request_enable = status.OSB

-- Clear the buffer and make it smaller
defbuffer1.clear()
defbuffer1.capacity = 100

-- Set the measure count to fill the buffer
dmm.measure.count = 100
dmm.measure.range = 10e-3
dmm.measure.read(defbuffer1)

printbuffer(1, defbuffer1.n, defbuffer1)
```

Index

KEITHLEY
A Tektronix Company

A Greater Measure of Confidence