# TTR500 Series
# Vector Network Analyzer

## Programmer Manual

077-1257-01

# TTR500 Series
# Vector Network Analyzer

## Programmer Manual

www.tek.com

077-1257-01

Copyright © Tektronix. All rights reserved. Licensed software products are owned by Tektronix or its subsidiaries or suppliers, and are protected by national copyright laws and international treaty provisions.

Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specifications and price change privileges reserved.

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

VectorVu-PC is a trademark of Tektronix, Inc.

## Contacting Tektronix

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tek.com to find contacts in your area.

# Table of Contents



## Get Started



## Syntax and Commands

## Status and Events

# Preface

When you open VectorVu-PC after connecting a TTR500 series Vector Network Analyzer (VNA) to your Windows machine, the software supports commands that control the functions of the VNA. This programmer manual contains information on operating the TTR500 VNA using VectorVu-PC vector analysis software. The manual applies to software version 1.2.4 and contains these sections:

- *Get Started* – How to send commands to the VNA.
- *Syntax and Commands* – Syntax definitions in the command descriptions and lists of all command subsystems.
- *Commands Descriptions* – Descriptions of all commands with examples.
- *Status and Events* – Operations of the Status and Events Reporting system, and a list of all system errors.

## Related Documentation

- *TTR500 Series Vector Network Analyzer Quick Start Guide*
  (Tektronix part number 071-3486-00, English)
  This manual contains general information about how to put your instrument into service.
- *TTR500 Series Vector Network Analyzer Help*
  This online help manual is configured in VectorVu-PC. It contains detailed information about how to operate the VNA.
- *TekVISA Programmer Manual*
  (Tektronix part number 077-0140-XX)
  This manual is available as a printable PDF file on the Tektronix web site (www.tek.com). The manual describes TekVISA, the Tektronix implementation of the VISA Application Programming Interface (API). TekVISA is industry-compliant software for writing interoperable instrument drivers in a variety of Application Development Environments (ADEs).

# Get Started

# Get Started

You can write computer programs that take measurements with the analyzer and read those measurements for further analysis or storage. These subsections contain information to help you get started with programming the analyzer:

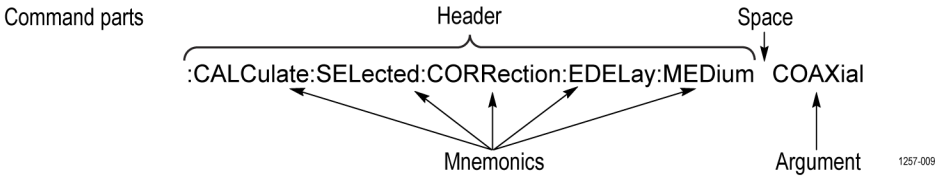
- *Overview of the Manual*
  Summarizes each major section of this manual.
- *Using TekVISA*
  Describes how to use the TekVISA communication protocol to communicate with the VNA to a programmatic interface.

# Overview of the Manual

The information contained in each major section of this manual is described here.

## Syntax and Commands

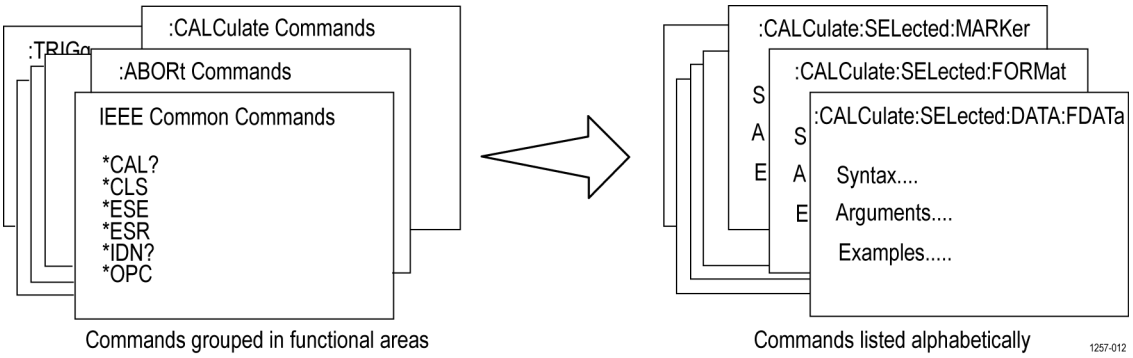
*Syntax and Commands* describes the structure and content of the messages your program sends to the analyzer. This figure shows command parts as described in the *Command Syntax* subsection:




The subsection *Command Groups* provides lists of all commands by functional areas.

## Command Descriptions

All commands are listed alphabetically in the *Command Descriptions* section. This section also describes the effect of each command and provides examples that use it.




## Status and Events

The program may request information from the instrument. The instrument provides information in the form of status and error messages. *Status and Events* describes how to get status or event information from the program and details the event and error messages.

# Install TekVISA

You must install the Tektronix Virtual Instrument Software Architecture (TekVISA) on your PC:

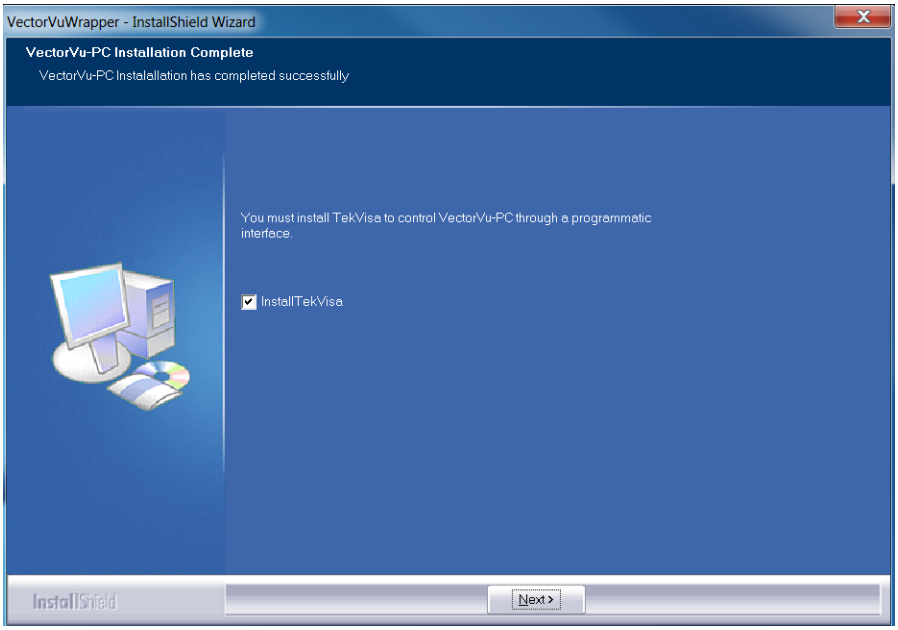
- To communicate to VectorVu-PC to a programmatic interface. TekVISA is the Tektronix implementation of VISA (Virtual Instrument Software Architecture), an industry-standard communication protocol.
- To perform power calibration using the TTR500 VNA

TekVISA is a collection of software and documentation that allows for remote control of instruments regardless of instrument brand and physical connection. You can use this software to write (or draw) interoperable instrument drivers in a variety of Application Development Environments (ADEs). It includes a VISA driver, VISA libraries, documentation, USB Device Driver, VXI-11 server, and connection management/debug software.

## Setup for the programming interface

The programming interface is automatically enabled as part of the VectorVu-PC application software installation. However, in order to control VectorVu-PC using the programmatic interface, you must also install TekVISA:

1. Install VectorVu-PC.
2. Once the software has installed, the InstallShield Wizard prompts you to install TekVisa as well. Proceed with this installation:




3. Install the Tektronix Power Meter applications next.
4. Click **Finish** to complete the installation.

## Enable control over the Ethernet

TekVISA supports communication with VectorVu-PC over an Ethernet LAN connection. The VXI-11 standard specifies an instrument protocol for TCP/IP computer networks, supporting writing and reading data to and from instruments.

The VXI-11 server is automatically installed on the target PC as part of the VectorVu-PC application software installation process. This also creates a shortcut in the Startup folder.

Right click the “TekVISA LAN Server Control” tray icon to set properties, check status, or start the VXI-11 or socket servers.

For more information about TekVISA concepts and operations, refer to the TekVISA Programmer Manual available at http://www.tek.com.

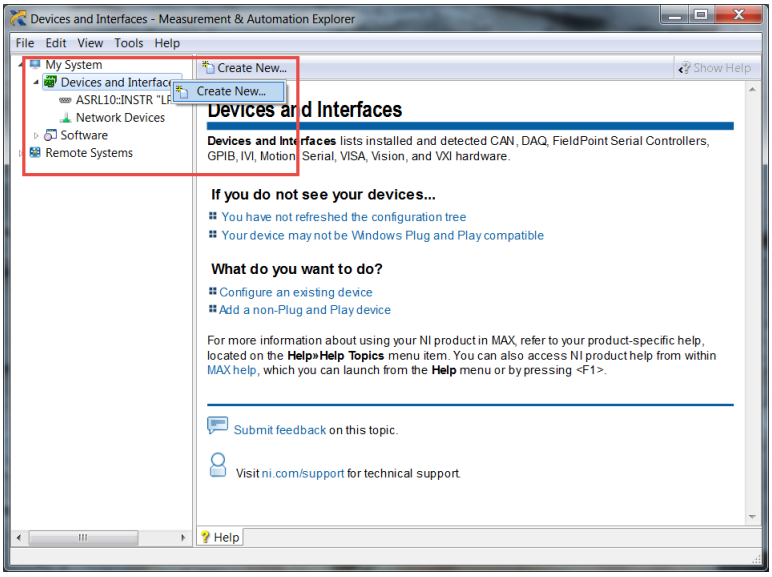
# Connect with NI-VISA and LabVIEW

LabVIEW users can operate the TTR500 VNA using the NI-VISA instrument manager from National Instruments. You can connect NI-VISA:

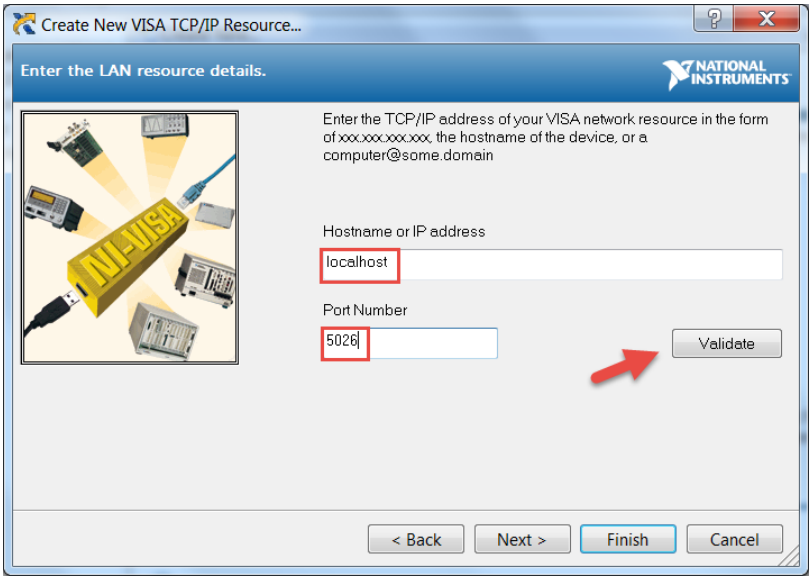
- Directly to VectorVu-PC (without TekVISA) to a socket connection. This option does not discover instrument connections automatically. Also, you can only pair one instance of VectorVu-PC with LabVIEW on a machine
- Through the TekVISA LAN Server Control using a VX-11 standard.

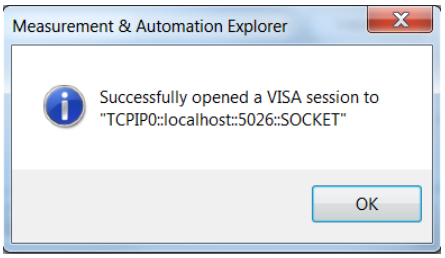
## Connect NI-VISA to VectorVu-PC without TekVISA

1. Open VectorVu-PC. Keep the program open to this procedure.
2. Open NI MAX.
3. Right click on **Devices and Interfaces**. Select the **Create New** option:



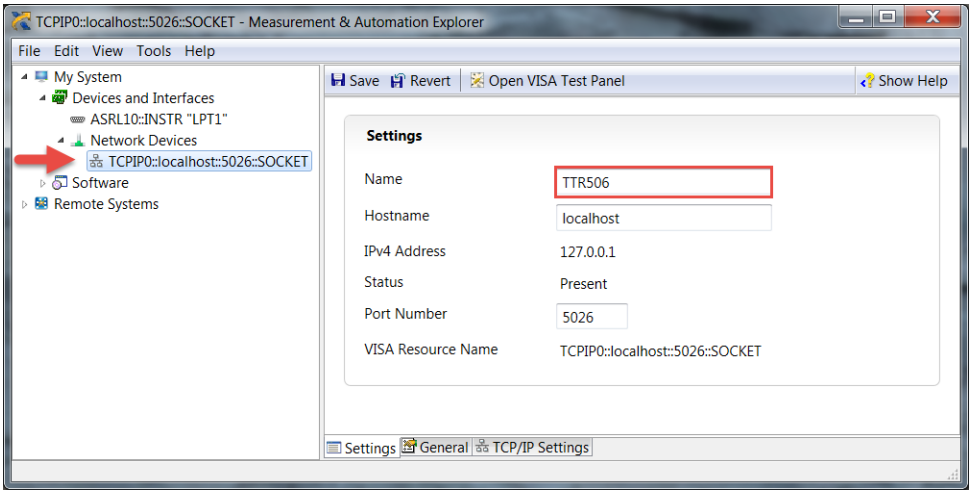

4. Select **Create new VISA TCP/IP Resource** and click **Next**.
5. In the next window, select **Manual Entry of Raw Socket**.
6. In the LAN resource details window, enter:
   - **Hostname**: localhost or 127.0.0.1
   - **Port number**: 5026
7. Click **Validate**. A popup window validates the connection:



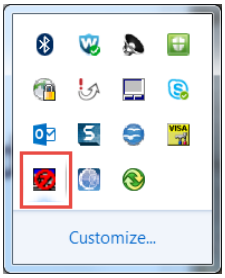

8. You can now finish the setup. The TTR500 device connection is displayed under **Network Devices** in the left pane.

9. Click on the TTR500 device connection. In the right pane, the **Settings** tab displays the connection information you entered. You can create an alias for this connection for future use:

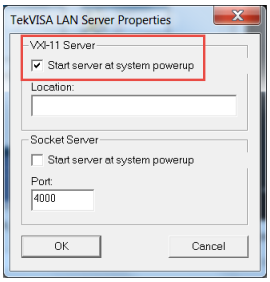


## Connect NI-VISA to Tek-VISA LAN Server Control

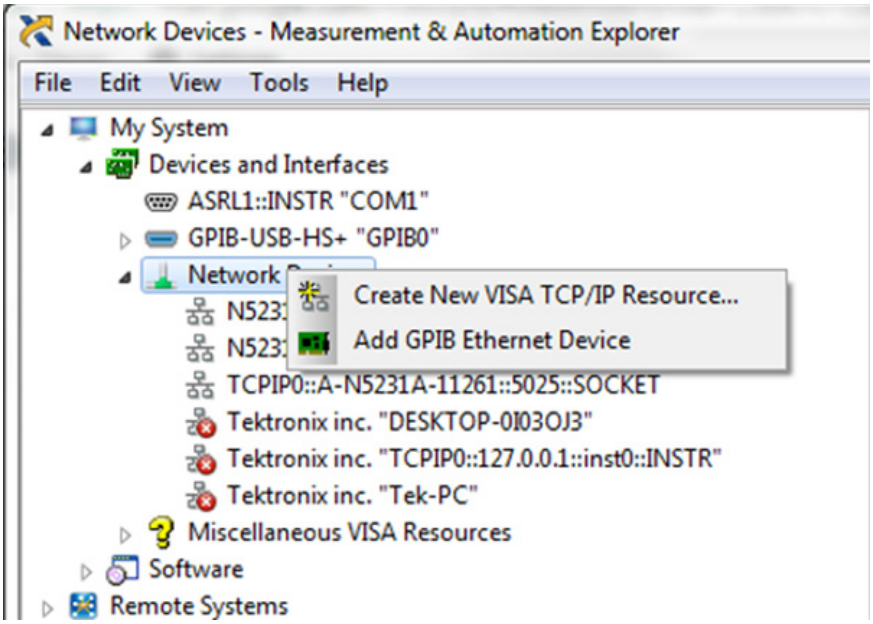
1. Click on the TekVISA LAN Server Control application. Open **Server Properties**.:



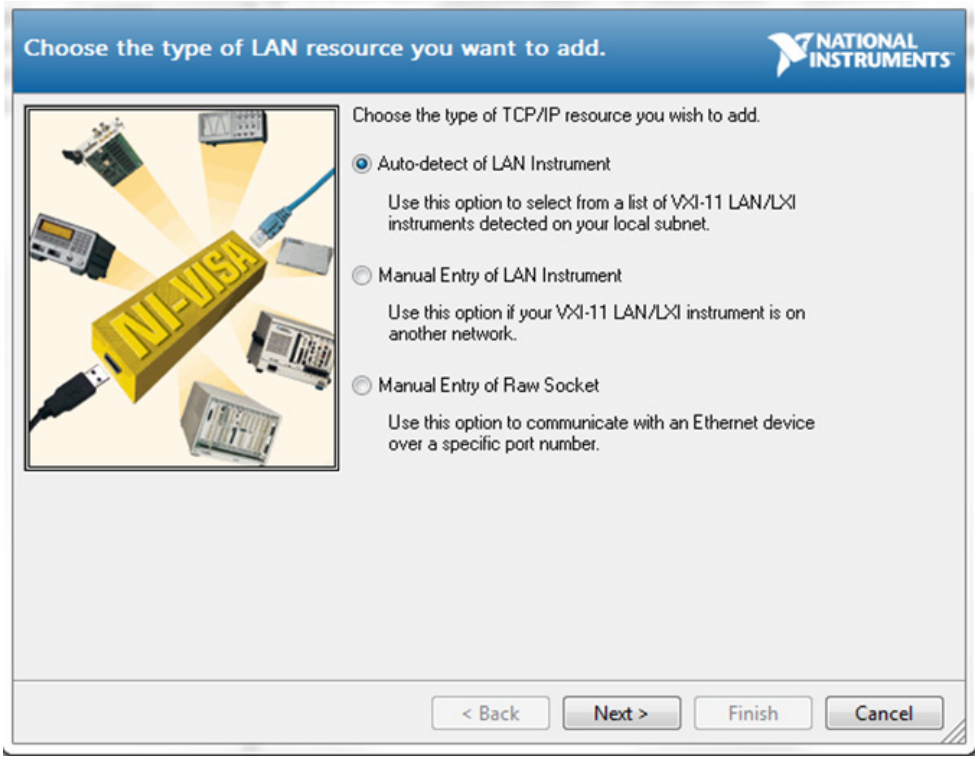

2. Under **VXI-11 Server**, check **Start server on system power up**:




3. Restart the machine.
4. Open NI-VISA and VectorVu-PC.
5. Right click **Network Devices** in NI-VISA and select **Create new VISA/TCP-IP Resource**:



6. Select **Auto-detect of LAN Instrument** and click **Next**:




NI-VISA should detect VectorVu-PC to enable the LabVIEW interface.

# Syntax and Commands

# Command Syntax

You can program the vector network analyzer (VNA) using commands and queries. The commands include Standard Commands for Programmable Instruments (SCPI) and IEEE 488.2 Common Commands. Commands and queries are organized in the following subsections:

- Backus-Naur Form Definition
- SCPI Commands and Queries
- IEEE 488.2 Common Commands
- Constructed Mnemonics

## Backus-Naur Form Definition

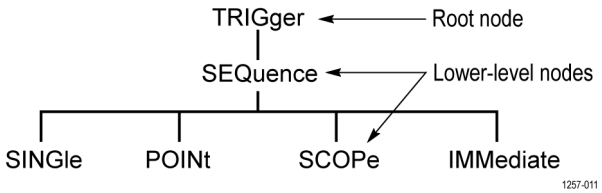
The definition for some commands and queries uses the Backus-Naur Form (BNF) notation. The following table defines standard BNF symbols:

| Symbol | Meaning |
|---|---|
| < > | Defined element |
| ::= | Is defined as |
| \| | Exclusive OR |
| { } | Group; one element is required |
| [ ] | Optional; can be omitted |
| . . . | Previous element(s) may be repeated |
| ( ) | Comment |

## SCPI Commands and Queries

SCPI is a standard created by a consortium that provides guidelines for remote programming of instruments. These guidelines provide a consistent programming environment for instrument control and data transfer. This environment uses defined programming messages, instrument responses, and data format across all SCPI instruments, regardless of manufacturer. The analyzer uses a command language based on the SCPI standard.

The SCPI language is based on a hierarchical or tree structure, which represents a subsystem, as shown in the following figure. The top level of the tree is the root node; it is followed by one or more lower-level nodes:




You can create commands and queries from these subsystem hierarchy trees. Commands specify actions for the instrument to perform. Queries return measurement data and information about parameter settings.

## Creating Commands

SCPI commands are created by stringing together the nodes of a subsystem hierarchy and separating each node by a colon.

In the above figure, TRIGger is the root node and SEQuence, IMMediate, POINt, SINGle, SCOPe and SOURce are all lower-level nodes. To create a SCPI command, start with the root node TRIGger and move down the tree structure adding nodes until you reach the end of a branch. Most commands and some queries have parameters and each parameter must include a value. If you specify a parameter value that is out of range, the parameter will be set to a default value. Command descriptions list the values that are valid for all parameters.

For example, TRIGger:SEQuence:SCOPe ALL is a valid SCPI command created from the hierarchy tree, using ALL as an argument.

## Creating Queries

To create a query, start at the root node of a tree structure and then move down to the end of a branch, nd add a question mark. For example, `TRIGger:SEQuence:SCOPe?` is a valid SCPI query, using the hierarchy tree in the figure.

## Query Responses

The query causes the analyzer to return information about its status or settings. When a query is sent to the analyzer, only the values are returned. For example, `SENSe1:AVERage:COUNt?` may return 400 as the averaging factor value for channel 1.

A few queries also initiate an operation action before returning information. For example, the `*CAL?` query runs a calibration.

## Parameter Types

Every parameter in the command and query descriptions is of a specified type. Parameters are enclosed in brackets, such as <value>. The parameter type is listed after the parameter and is enclosed in parentheses, for example, (boolean). Some parameter types are defined specifically for the VNA Series command set and some are defined by ANSI/IEEE 488.2-1987 as shown on the following table:

| Parameter type | Description | Example |
|---|---|---|
| arbitrary block [1] | Specified length of arbitrary data | #512234xxxxx . . . where 5 indicates that the following 5 digits (12234) specify length of data in bytes; xxxxx ... indicates data |
| boolean | Boolean numbers or values | ON or 1; OFF or 0 |
| binary | Binary numbers | #B0110 |
| octal | Octal numbers | #Q57, #Q3 |
| hexadecimal [2] | Hexadecimal numbers (0-9, A, B, C, D, E, F) | #HAA, #H1 |
| NR1 [2] numeric | Integers | 0, 1, 15, -1 |
| NR2 [2 3] numeric | Decimal numbers | 1.2, 3.141516, -6.5 |
| NR3 [2] numeric | Floating point numbers | 3.1415E-9, -16.1E5 |
| NRf [2] numeric | Flexible decimal number may be type NR1, NR2 or NR3 | See NR1, NR2, and NR3 examples |
| string [4] | Alphanumeric characters (must be within quotation marks) | "Testing 1, 2, 3" |

1 Defined in ANSI/IEEE 488.2 as Definite Length Arbitrary Block Response Data.

2 An ANSI/IEEE 488.2-1992-defined parameter type.

3 Some commands and queries will accept an octal or hexadecimal value even though the parameter type is defined as NR1.

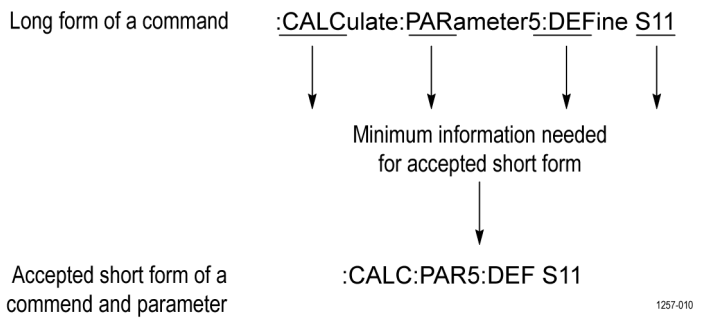
4 Defined in ANSI/IEEE 488.2 as String Response Data.

## Special Characters

The Line Feed (LF) character (ASCII 10), and all characters in the range of ASCII 127-255 are defined as special characters. These characters are used in arbitrary block arguments only; using these characters in other parts of any command yields unpredictable results.

## Abbreviating Commands, Queries, and Parameters

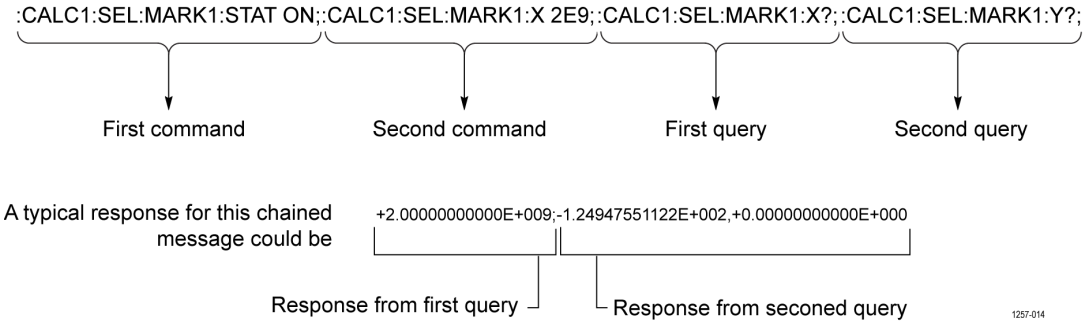
You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these short forms as a combination of upper and lower case letters. The upper case letters indicate the accepted short form of a command. You can create a short form by using only the upper case letters, as shown in the next figure. The accepted short form and the long form are equivalent and request the same action of the instrument:



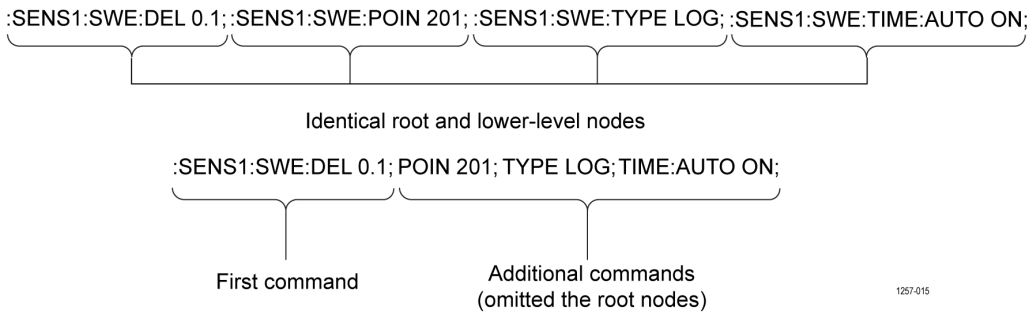

***NOTE.*** *The numeric suffix of a command or query may be included in either the long form or short form; the analyzer will default to "1" if no suffix is used. In the above figure, the "5" in "PARameter5" indicates that the command is targeting trace 5 for channel 1.*

## Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, add a semicolon (;), and then add more commands or queries and semicolons until the message is complete. Commands following a semicolon are a root node. You must precede them with a colon ":". The following figure illustrates a chained message consisting of several commands and queries. The single chained message must end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.




If a command or query have the same root and lower-level nodes as the previous command or query, you can omit these nodes. In the following figure, the second, third, and fourth commands have the same root node (SENSe1:SWEep) as the first command, so these nodes can be omitted.




## Unit and SI Prefix

If the decimal numeric argument refers to amplitude, frequency, or time, you can express it using SI units instead of using the scaled explicit point input value format <NR3>. (SI units are units that conform to the Systeme International d'Unites standard.) For example, you can use the input format 200 mV or 1.0 MHz instead of 200.0E-3 or 1.0E+6, respectively, to specify voltage or frequency. The next table lists the available units:

| Symbol | Meaning |
|---|---|
| dB | Decibel (relative amplitude) |
| dBm | Decibel (absolute amplitude) |
| DEG | Degree (phase) |
| Hz | Hertz (frequency) |
| PCT | Percent (%) |
| s | Second (time) |
| V | Volt |

The available SI prefixes are shown in the following table.

| SI prefix | Z | A | F | P | N | U | M | K | MA[1] | G | T | PE | EX |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Corresponding power | $10^{-21}$ | $10^{-18}$ | $10^{-15}$ | $10^{-12}$ | $10^{-9}$ | $10^{-6}$ | $10^{-3}$ | $10^{+3}$ | $10^{+6}$ | $10^{+9}$ | $10^{+12}$ | $10^{+15}$ | $10^{+18}$ |

1 When the unit is "Hz", "M" may be used instead of "MA" so that the frequency can be represented by "MHz".

You can omit a unit in a command, but you must include the unit when using a SI prefix. For example, a frequency of 15 MHz can be described:

15.0e6, 1.5e7 Hz, 15000000, 15000000 Hz, 15 MHz, etc.

("15M" is not allowed.)

Note that you can use either lower or upper case units and prefixes. The following examples have the same result, respectively.

170 mHz, 170 Mhz, 170 MHz, etc.

250 mv, 250 mV, 250 MV, etc.

## General Rules

Here are three general rules for using SCPI commands, queries, and parameters:

- You can use single (‘ ’) or double (“ ”) quotation marks for quoted strings, but you cannot use both types of quotation marks for the same string.

  correct "This string uses quotation marks correctly."

  correct ‘This string also uses quotation marks correctly.'

  incorrect "This string does not use quotation marks correctly.'

- You can use upper case, lower case, or a mixture of both cases for all commands, queries, and parameters.

  ```
  SYSTEM:BEEPER:WARNING:STATE ON
  ```

  is the same as

  ```
  system:beeper:warning:state on
  ```

  and

  ```
  SYSTem:BEEPer:WARNing:STATe ON
  ```

***NOTE.*** *Literal strings (quoted) are case sensitive, such as file names.*

- No embedded spaces are allowed between or within nodes:

  correct `SYSTem:BEEPer:WARNing:STATe ON`

  incorrect `SYSTem:  BEEPer:  WARNing:  STATe ON`

## IEEE 488.2 Common commands

**Description** ANSI/IEEE Standard 488.2 defines the codes, formats, protocols, and usage of common commands and queries used on the interface between the controller and the instruments. The analyzer complies with this standard.

### Command and Query Structure

The syntax for an IEEE 488.2 common command is an asterisk (*) followed by a command and, optionally, a space and parameter value. The syntax for an IEEE 488.2 common query is an asterisk (*) followed by a query and a question mark. All of the common commands and queries are listed in the last part of the *Syntax and Commands* section. These are examples of common commands:

- *ESE 16
- *CLS

These are examples of common queries:

- *ESR
- *IDN

## Constructed mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic can be either CHANnel1, CHANnel2, or a different number (1–16). Use these mnemonics in the command just as you do any other mnemonic. For example, there is a SENSe1:CORRection:COEFficient:DATA command, and there is also a SENSe2:CORRection:COEFficient:DATA command, where the first applies to channel 1 (hence SENSe1) and the second applies to channel 2 (hence SENSe2).

In command descriptions, this list of choices is abbreviated as SENSe<x>, where *x* refers to the channel number for SENSe commands. The value of <x> is the upper range of valid suffixes. If the numeric suffix is omitted, the analyzer uses the default value of "1".

| Item | Description |
|---|---|
| MARKer<x> | Marker specifier where <x> = 1–9<br>See *Marker Mnemonics*. |
| TRACe<x> | Trace specifier where <x> = 1–16 |
| CHANnel<x> | Channel specifier where <x> = 1–16 |

# Command groups

This section lists the analyzer commands in two ways:

- Functional groups that are based on the action performed.
- Alphabetical list providing more detail about each command.

VNA series analyzers conform to the Standard Commands for Programmable Instruments (SCPI) 1999.0 and IEEE Std 488.2-1987 except where noted.

Items followed by question marks are queries and items without question marks are commands. Some items in this section have a question mark in parentheses "( )" in the command header section; this indicates that the item can be both a command and a query.

For the conventions of notation in this manual, refer to *Command Syntax* and following pages.

## Functional groups

The SCPI commands for the TTR500 series VNA are divided into these functional groups, based on the action they perform:

| Group | Function |
|---|---|
| ABORt | Resets trigger system and stops measurements. |
| CALCulate | Controls markers and search operations. |
| DISPlay | Controls display of measurement results. |
| FORMat | Controls formats where measurement results show. |
| IEEE common | Conforms to IEEE Std 488.2. |
| INITiate | Controls data acquisition. |
| MMEMory | Provides mass storage capabilities for analyzer. |
| OUTPut | Controls signal output characteristics. |
| SENSe | Sets detailed conditions for all measurements. |
| STATus | Controls status and event registers. |
| SERVice | Controls analyzer service operations. |
| SIMulator | Controls analyzer in simulator offline mode. |
| SOURce | Controls power calibration function. |
| STATus | Controls status event register operations. |
| SYStem | Controls beep, warnings, and errors. |
| TRIGger | Controls triggering function. |

## IEEE common commands

The IEEE 488.2 common commands have a "*" prefix.

| Command | Description |
|---|---|
| *WAI | Clears statuses. |
| *ESE | Sets or queries ESER register bits. |
| *ESR? | Returns SESR register content. |
| *IDN? | Returns instrument identification code. |
| *OPC | Synchronizes commands. |
| *RST | Returns instrument settings to factory defaults. |
| *SRE | Sets or queries bits in SRER register. |
| *STB? | Returns SBR contents using MSS bit. |
| *WAI | Prevents analyzer from executing further commands. |

# Abort commands

Use Abort commands to stop current measurements for all channels.

| Command | Description |
|---|---|
| ABORt | Abort current measurement for all channels. |

# Calculate commands

Use the Calculate commands to control markers on the active trace of a channel.

| Command | Description |
|---|---|
| CALCulate<x>:PARameter:COUNt | Set or query the number of traces on the specified channel. |
| CALCulate<x>:PARameter<y>:DEFine | Set or query a measurement parameter of the specified trace on the specified channel. |
| CALCulate<x>:PARameter<y>:[SELect] | Set the selected trace as the active trace on the specified channel. |
| CALCulate<x>:PARameter<y>:RPORt | Set or query the receiver port of the selected trace on the specified channel. |
| CALCulate<x>:PARameter<y>:SPORt | Set or query the source port of the specified trace on the specified channel. |
| CALCulate<x>:[SELected]:BLIMit:DB | Set or query the bandwidth threshold value (dB) for the bandwidth test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:BLIMit:DISPlay:MARKer | Set or query the state of the display for the marker of the bandwidth test for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:BLIMit:DISPlay:VALue | Set or query the state of the display for the bandwidth value of the bandwidth test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:BLIMit:FAIL? | Query the result for the bandwidth test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:BLIMit:MAXimum | Set or query the upper limit value for the bandwidth test of the active trace on the specified channel. When the upper limit value is set to power sweep, the maximum and minimum values go outside the dB range. |
| CALCulate<x>:[SELected]:BLIMit:MINimum | Set or query the lower limit value (in Hz) for the bandwidth test in the active trace for the specified channel. |
| CALCulate<x>:[SELected]:BLIMit:REPort:DATA? | Query the bandwidth value for the bandwidth test of the active trace on the specified channel. The bandwidth test must be enabled using CALCulate<x>:[SELected]:BLIMit:[STATe] to properly use this command. The bandwidth test returns a value in power sweep. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:BLIMit:[STATe] | Set or query the bandwidth limit test for the active trace for the specified channel. |
| CALCulate<x>:[SELected]:CORRection:EDELay:MEDium | Set or query the medium for the electrical delay on the specified channel. |
| CALCulate<x>:[SELected]:CORRection:EDELay:TIME | Set or query the time for the electrical delay for the active trace on the specified channel. If the specified delay time is outside the range, the minimum or maximum time is set accordingly. |
| CALCulate<x>:[SELected]:CORRection:EDELay:WGCutoff | Set or query the cutoff frequency (in Hz) when a waveguide delay is used for the electrical delay on the specified channel. If the specified frequency is outside the range, the minimum or maximum frequency is set accordingly. |
| CALCulate<x>:[SELected]:CORRection:OFFSet:PHASe | Set or query the phase offset of the active trace on the specified channel. If the specified phase offset is outside the range, the minimum or maximum offset is set accordingly. |
| CALCulate<x>:[SELected]:DATA:FDATa | Set or query the formatted data array of the active trace on the specified channel. To set this command, you must provide two times the number of points as the active trace. This is to account for both main and auxiliary values. |
| CALCulate<x>:[SELected]:DATA:FMEMory? | Query the formatted memory array of the active trace of on specified channel. |
| CALCulate<x>:[SELected]:DATA:SDATa | Set or query the corrected data array of the active trace on the specified channel. To set this command, you must provide two times the number of points as the active trace. |
| CALCulate<x>:[SELected]:DATA:SMEMory? | Query the corrected memory array for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:DATA:RDATa | Set or query the raw data array for the active trace on the specified channel. To set this command, you must set the same number of points as the number of points contained on the active trace. |
| CALCulate<x>:[SELected]:FORMat | Set or query the data format of the active trace on the specified channel. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:FUNCtion:DATA? | The query returns an alphanumeric value for the analysis results obtained from CALCulate<x>:[SELected]:FUNCtion: EXECute based on the analysis set by CALCulate<x>:[SELected]:FUNCtion: TYPE for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:DOMain: COUPle | Set or query the analysis range to all traces on the specified channel. The query returns whether the analysis range to all traces on the specified channel is decoupled (0) or coupled (1). |
| CALCulate<x>:[SELected]:FUNCtion:DOMain: STARt | Set or query the starting stimulus value for the arbitrary analysis range (in Hz) used by CALCulate<x>:[SELected]:FUNCtion: DOMain:STARt for the specified channel. If the analysis range is coupled, the range is set for all traces on the specified channel. Otherwise, the active trace start value is set instead. |
| CALCulate<x>:[SELected]:FUNCtion:DOMain: [STATe] | Set or query whether to use an arbitrary range or an entire sweep range on the specified channel, when executing the analysis with CALCulate<x>:[SELected]:FUNCtion: EXECute. |
| CALCulate<x>:[SELected]:FUNCtion:DOMain: STOP | Set or query the stopping stimulus value (in Hz) for the arbitrary analysis range used by CALCulate<x>:[SELected]:FUNCtion: EXECute on the specified channel. If the analysis range is coupled, the range is set for all traces on the specified channel. Otherwise, the active trace stopping value is set instead. |
| CALCulate<x>:[SELected]:FUNCtion:EXECute | Execute the statistical analysis set by CALCulate<x>:[SELected]:FUNCtion: TYPE for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:PEXCursion | Set or query the lower limit (in Hz) of the peak excursion value when performing a peak search using CALCulate<x>:[SELected]:FUNCtion: EXECute for the active trace on the specified channel. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:FUNCtion:POINts? | Queries the number of data pairs found on the analysis result when using CALCulate<x>:[SELected]:FUNCtion: EXECute for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:PPOLarity | Set or query the polarity of the peak excursion search when performing a peak search using CALCulate<x>:[SELected]:FUNCtion: EXECute for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:TARGet | Set or query a target value (in Hz) for the target search when using CALCulate<x>:[SELected]:FUNCtion: EXECute for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:TTRansition | Set or query the transition type for the target search when using CALCulate<x>:[SELected]:FUNCtion: EXECute for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:TYPE | Set or query the analysis type when using CALCulate<x>:[SELected]:FUNCtion: EXECute for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:[STATe] | Set or query the status of the limit test for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:DATA | Set or query the limit table for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:DISPlay:[STATe] | Set or query the state of the display for the limit line defined in CALCulate<x>:[SELected]:LIMit:DATA for the enabled limit test set by CALCulate<x>:[SELected]:LIMit:[STATe] for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:FAIL? | Query the limit test result for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:OFFSet: AMPLitude | Set or query the amplitude offset value (in Hz) for the limit line test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:OFFSet:MARKer | Set the amplitude offset value to the active marker value for the limit line test of the active trace on the specified channel. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:LIMit:OFFSet:STIMulus | Set or query the stimulus offset value (in Hz) for the limit line test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:REPort:ALL? | Query the limit test result (in Hz) for all measurement points of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:REPort:DATA? | Query the stimulus values (in Hz) of all measurement points that failed the limit test for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:LIMit:REPort:POINts? | Query the number of measurement points that failed the limit test for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:[STATe] | Set or query the state of the display for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:[ACTivate] | Set the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:BWIDth:DATA? | Query the result value(s) (in Hz) for the bandwidth search of the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:BWIDth:THReshold | Set or query the bandwidth definition value (in Hz) for the specified marker of the active trace on the specified channel |
| CALCulate<x>:[SELected]:MARKer<y>:BWIDth:[STATe] | Set or query the state of the display of the bandwidth search result for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:COUPle | Set or query state of the marker coupling for the specified marker between traces on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:DISCrete | Set or query the state of the marker discrete mode for the specified marker of the active trace of on the specified channel. In the discrete mode, the marker moves only at measurement points. While in continuous mode, the marker will interpolate data between measurement points. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe] | Set or query an arbitrary or entire sweep range for a marker search on the specified marker of the active trace of on the specified channel. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle | Set or query the state of the marker search range (coupled or decoupled) for the specified marker to all traces on a marker search for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt | Set or query the starting value (in Hz) for the arbitrary marker search range for a marker search on the specified marker for the active trace on the specified channel. The arbitrary marker search range is set with CALCulate<x>:[SELected]:MARKer:FUNCtion:DOMain:[STATe]. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP<br>! | Set or query the stopping value for the arbitrary marker search range for the specified marker of the active trace for the specified channel. |
| CALCulate<x>:[SELected]:FUNCtion:DOMain:STOP | Set or query the stopping value (in Hz) for the arbitrary marker search range for a marker search for the specified marker of the active trace on the specified channel. |
| CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute | Execute the marker search set by CALCulate<x>:[SELected]:MARKer:FUNCtion:TYPE for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PEXCursion | Set or query the lower limit for the peak excursion value when executing a peak excursion search for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PPOLarity | Set or query the polarity for a peak excursion search for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TARGet | Set or query a target value (in Hz) for a target search on the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TRACking | Set or query the state of the marker tracking for a marker search, using the specified marker of the active trace on the specified channel. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TTRansition | Set or query the transition type used during a target search for the specified marker on the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE | Set or query the type of search used in CALCulate<x>:[SELected]:FUNCtion:EXECute for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:NOTCh:DATA? | Query the results from a notch search using the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:NOTCh:THReshold | Set or query the notch value definition during a notch search, using the specified marker of the active trace of on specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:NOTCh:[STATe] | Set or query the notch search for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:SET | Set the value for the specified marker position to the specified value of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:X | Set or query the marker stimulus value (in Hz) for the specified marker on the active trace on the specified channel. If the specified frequency is outside the range, the minimum or maximum frequency is set accordingly. |
| CALCulate<x>:[SELected]:MARKer:Y? | Query the marker response value for the specified marker of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:MARKer<y>:REFerence:[STATe] | Set or query the state of the display for the reference marker mode of the specified marker of the active trace on the specified channel. If the reference marker is not enabled, this command will then enable the reference marker. |
| CALCulate<x>:[SELected]:MATH:FUNCtion | Set or query the data trace method (math operation) for the data trace based on the stored memory trace CALCulate<x>:[SELected]:MATH:MEMorize for the active trace on the specified channel. Display the math result on the data trace. |
| CALCulate<x>:[SELected]:MATH:MEMorize | Copy the measurement data to the memory trace of the active trace on the specified channel. |

| Command | Description |
|---|---|
| CALCulate<x>:[SELected]:RLIMit:DATA | Set or query the value for the ripple limit table for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:RLIMit:DISPlay:LINE | Set or query the state of the display of the ripple limit line for the active trace on the specified channel, by setting both the ripple limit test CALCulate<x>:[SELected]:RLIMit:[STATe] and ripple limit table CALCulate<x>:[SELected]:RLIMit:DATA. |
| CALCulate<x>:[SELected]:RLIMit:DISPlay:[SELect] | Set or query the ripple limit band that shows the value of the ripple test result for the active trace on the specified channel. Set the display value for the ripple limit band with CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue. |
| CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue | Set or query the display type for the ripple value of the active trace on the specified channel. Change between ripple lines to display the ripple value with CALCulate:[SELected]:RLIMit:DISPlay:[SELect]. |
| CALCulate<x>:[SELected]:RLIMit:FAIL? | Query all ripple value results for the ripple limit test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:RLIMit:REPort:DATA? | Query all ripple value results for the ripple limit test of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:RLIMit:[STATe] | Set or query the state of the display of the ripple test for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:SMOOthing:APERture | Set or query the smoothing aperture percentage for the sweep span of the active trace on the specified channel. |
| CALCulate<x>:[SELected]:SMOOthing:[STATe] | Set or query the smoothing aperture for the active trace on the specified channel. |
| CALCulate<x>:[SELected]:PHASe | Set or query the phase unit used for data formats PHASe, UPHase, and PPHase (CALCulate<x>:[SELected]:FORMat) for the active trace on the specified channel. |

## Marker Mnemonics

You can use 9 markers and a reference marker in the VNA series analyzers. In commands, these are named MARKer<x>, where <x> can be 1–9.

| Mnemonic | Description |
|---|---|
| R | Reference marker (MR) |
| MARKer1 | Marker 1 (M1) |
| MARKer2 | Marker 2 (M2) |
| MARKer3 | Marker 3 (M3) |
| MARKer4 | Marker 4 (M4) |
| MARKer5 | Marker 5 (M5) |
| MARKer6 | Marker 6 (M6) |
| MARKer7 | Marker 7 (M7) |
| MARKer8 | Marker 8 (M8) |
| MARKer9 | Marker 9 (M9) |

***NOTE.*** *If you omit the numeric suffix, the marker control defaults to Marker 1.*

*Before operating the marker, you have to enable it using the CALCulate basic commands.*

*If you attempt to use a marker other than the above markers in a CALCulate command, the suffix error (error code -130) occurs.*

# Display commands

Use DISPlay commands to control the display of measurement results.

| Command | Description |
|---|---|
| DISPlay:ANNotation:FREQuency:[STATe] | Set or query the state of the display of the frequency for all windows. |
| DISPlay:CHANnel<x>:[ACTivate] | Set the channel for the active graph set by DISPlay:GRAPh:[ACTivate]. |
| DISPlay:CHANnel<x>:ANNotation:MARKer:ALIGn:[STATe] | Set or query the status of the alignment for all marker readouts for markers of trace 1 on the specified channel. |
| DISPlay:CHANnel<x>:ANNotation:MARKer<y>:SINGle:[STATe] | Set or query the status of the display of the specified marker information for all traces or only for the active trace on the specified channel. |
| DISPlay:CHANnel<x>:LABel | Set or query the state of the display of the vertical axis label on the specified graph and channel. |
| DISPlay:MAXimize | Set or query the maximization status for the active trace on the specified channel. |
| DISPlay:CHANnel<x>:SPLit | Set or query the display layout for traces on the selected channel. |
| DISPlay:CHANnel<x>:TITLe:DATA | Set or query the label title for the specified channel. |
| DISPlay:CHANnel<x>:TITLe:[STATe] | Set or query the state of the display of the label title for the specified channel. |
| DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:X | Set or query the display position of the marker value on the X-axis, using a percentage of the width of the display span for the specified trace on the specified channel. |
| DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:Y | Set or query the position of the marker value on the Y-axis, using a percentage of the width of the display span for the specified trace on the specified channel. |
| DISPlay:CHANnel<x>:TRACe<y>:MEMory:[STATe] | Set or query the status for the display of the memory trace for the specified trace on the specified channel. To store the memory trace of the specified trace, use CALCulate<x>:[SELected]:MATH:MEMorize. |
| DISPlay:CHANnel<x>:TRACe<y>:[STATe] | Set or query the status of the display for the data trace on the specified trace for the specified channel. |

| Command | Description |
|---|---|
| DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:AUTO | Execute the auto scale function for the specified trace on the specified channel. This function adjusts the value of the reference value and the scale per division to display the trace. |
| DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:PDIVision | Set or query the scale per division of the specified trace on the specified channel, when the data format is not Smith or Polar format. Set the full scale value (value of the outermost circumference), if the data format is Smith Chart or Polar formats. If the specified value is outside the allowable setup range, a minimum or maximum value is set. |
| DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RLEVEL | Set or query the value of the reference division line on the specified trace for the specified channel. |
| DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RPOSition | Set or query the position of the reference division line, using its position number (assigned integer starting from the lowest division) on the specified trace for the specified channel. |
| DISPlay:CHANnel<x>:X:SPACing | Set or query the type of display of the graph horizontal axis on the specified channel. |
| DISPlay:CHANnel<x>:Y:[SCALe]:DIVisions | Set or query the number of divisions in all graphs on the specified channel. |
| DISPlay:GRAPh<x>:[ACTivate]:CHANnel | Set the specified channel to the specified graph, only if the channel layout displays the corresponding specified graph value and the specified channel value is enabled. |
| DISPlay:GRAPh<x>:[ACTivate]:LABel | Set to display or hide the Y-axis label on the specified graph. |
| DISPlay:ENABle | Set or query the state of the display of the update on the graph area. |
| DISPlay:FONT:[BAR] | Set or query the menu bar text font size. |
| DISPlay:FONT:GRAPh | Set or query the graph text font size. |
| DISPlay:FSIGn | Set or query the display of the fail sign for the limit, ripple limit, and bandwidth tests. |
| DISPlay:GRAPh<x>:[ACTivate]:CHANnel | Set the specified channel to the specified graph, only if the channel layout displays the corresponding specified graph value and the specified channel value is enabled. |

| Command | Description |
|---|---|
| DISPlay:GRAPh<x>:[ACTivate]:LABel | Set to display or hide the Y-axis label on the specified graph. |
| DISPlay:GRAPh<x>:[ACTivate]:TITLe:DATA | Set the title of the specified graph. |
| DISPlay:GRAPh<x>:[ACTivate]:TITLe:[STATe] | Set to display or hide the channel title for the specified graph. |
| DISPlay:GRAPh<x>:[ACTivate] | Set the specified graph as the active graph. The channel layout display setting must allow for the graph number of this command to work. |
| DISPlay:GRAPh<x>:[ACTivate]:TRACes | Set or query the specified traces to the specified graph for the active channel. |
| DISPlay:IMAGe | Set or query the color scheme for the display. |
| DISPlay:MAXimize | Set or query maximization for the active channel. |
| DISPlay:MENU:[STATe] | Set or query the status of the display of the menu bar. |
| DISPlay:WINDow<x>:[ACTivate] | Set the specified channel as the active channel of the active graph. The channel must be enabled to allow the specified channel to be set as the active channel. |
| DISPlay:SPLit | Set or query the display layout for the channels. |
| DISPLay:TABLe:[STATe] | Set or query the status of the display window located on the lower part of the screen. |
| DISPlay:TABLe:TYPE | Set or query the type of window displaying the lower part of the screen. |
| DISPlay:CHANnel<x>:TRACe<y>:[STATe] | Set or query the status for the display of the data trace for the specified trace on the specified channel. |
| DISPlay:UPDate:[IMMediate] | Execute a single, immediate update of the display when DISPlay:ENABle has set updates to 0. |

# Initiate commands

Use INITiate commands to control the acquisition of data.

| Command | Description |
|---|---|
| INITiate<x>:CONTinuous | Set or query the status of the trigger mode (continuous or hold) that is used for the specified channel. |
| INITiate<x>:[IMMediate] | Execute a single sweep for the specified channel, while the channel is being set to hold. After executing the measurement once, the trigger system for the channel returns to an idle state. When you execute this command for a channel that is not idle (with hold or continuous mode enabled), an error occurs. |

# Mass Memory Commands

Use MMEMory commands to manipulate files on the mass memory devices.

| Command | Description |
|---|---|
| MMEMory:LOAD:ASCFactor | Load the power sensor calibration factor table for the specified port from the specified CSV file for the active channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:LOAD:CKIT | Load the definition of the specified calibration kit from the specified CSV file. Make sure the CSV file is available at the provided path as no notification is provided if the file is not loaded. |
| MMEMory:LOAD:LIMit | Load the limit table from the specified CSV file for the active trace. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:LOAD:PROSs | Load the information contained in the specified CSV file for the loss compensation table of the power calibration in the active channel of the specified port. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:LOAD:RLIMit | Load the ripple limit table from the specified CSV file for the active trace. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:LOAD:SEGMent | Load the segment sweep table for the active channel from the specified CSV file. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:LOAD:[STATe] | Load the specified instrument state file. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |

| Command | Description |
|---|---|
| MMEMory:STORe:PSCFactor | Save the power sensor calibration factor table for the specified port in the specified CSV file for the active channel. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:CKIT | Save the definition of the specified calibration kit in the specified CSV file for the active channel. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:LIMit | Save the limit table in the specified CSV file for the active trace. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:RLIMit | Save the ripple limit table for the active trace in the specified CSV file. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:SALL | Set or query what content (all channels/traces or only displayed channels/traces) must be saved as the instrument state. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:SEGMent | Store the segment sweep table for the active channel from the specified CSV file. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:PLOSs | Save the loss compensation table for power calibration specified in the CSV file for the specified port. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |

| Command | Description |
|---|---|
| MMEMory:STORe:SNP:[DATA] | Save the measurement data as a touchstone file in SnP format (S1P or S2P). The measurement data format is applied by MMEMory:STORe:SNP:FORMat. The SnP format is set by MMEMory:STORe:SNP:TYPE:S1P or MMEMory:STORe:SNP:TYPE:S2P. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. The SnP format is not automatically appended to the file name. |
| MMEMory:STORe:SNP:FORMat | Set or query the data format used to save measurement data in a touchstone file for the active channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded. The SnP format is not automatically appended to the file name. |
| MMEMory:STORe:SNP:NPORTS? | Query the number of ports used when saving a touchstone file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded. The SnP format is not automatically appended to the file name. |
| MMEMory:STORe:SNP:PORTS? | Query the specified ports used when saving a touchstone file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded. The SnP format is not automatically appended to the file name. |
| MMEMory:STORe:STYPe? | Query the last extension used to save the instrument's state file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded. |
| MMEMory:STORe:SNP:TYPE:S1P | Set the port used for the S1P file type when saving measurement data for the active channel to touchstone format. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. The SnP format is not automatically appended to the file name. |

| Command | Description |
|---|---|
| MMEMory:STORe:SNP:TYPE:S2P | Set the ports used for the S2P file type when saving measurement data to touchstone format for the active channel. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. The SnP format is not automatically appended to the file name. |
| MMEMory:STORe:[STATe] | Save the state of the instrument to the specified state file. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded. |

## Specifying the File

For loading and storing a file, follow these rules:

- You can omit the file extension to load and store data. The measurement-specific extension is automatically added.
- You can use the absolute path to specify the file name. For example, specify the *SAMPLE1* file in the *My Documents* folder on the C drive as `"C:\My Documents\SAMPLE1"`.
- If you omit the directory path, the default path is used, which is *C:\Program Files\Tektronix\VectorVu-PC* initially.

Once a file is saved to a different directory, the new directory will be used as the default for all load and store operations.

# Output Commands

Use OUTPut commands to control the characteristics of the signal output.

| Command | Description |
|---|---|
| OUTPUT:[STATe] | Set or query the stimulus signal output. You can only make measurements when the signal output is enabled. |

# Sense commands

Use SENSe commands to set up detailed conditions for each measurement.

| Command | Description |
|---|---|
| SENSe<x>:AVERage:CLEar | Reset the data count to 0 when using the averaging function set by SENSe<x>:AVERage:[STATe] for the specified channel. Measurement data recorded before executing this object is not used for averaging. |
| SENSe<x>:AVERage:COUNt | Set or query the averaging factor (number of traces to average) for the selected channel. |
| SENSe<x>:AVERage:[STATe] | Set or query the averaging function for the selected channel. |
| SENSe<x>:BANDwidth:RESolution | Set or query the IF bandwidth for the specified channel. |
| SENSe<x>:CORRection:CLEar | Clear the calibration error coefficients for the specified channel. |
| SENSe<x>:CORRection:COEFficient:[DATA] | Set or query the calibration coefficient data for the specified channel. When the calibration factor is interpolated, the interpolated calibration coefficient array is read. The written value in the coefficient array becomes effective only after you save the coefficient array using SENSe<x>:CORRection:COEFficient: SAVE. |
| SENSe<x>:CORRection:COEFficient:HDR | Set or query the HDR gain path for each port for the specified channel for the purpose of error terms. |
| SENSe<x>:CORRection:COEFficient:METHod: ERESPonse | Set the calibration type to enhanced response calibration for the specified channel. |
| SENSe<x>:CORRection:COEFficient:METHod: RESPonse:OPEN | Set the calibration type for the specified channel to the response calibration open for the specified port. |
| SENSe<x>:CORRection:COEFficient:METHod: RESPonse:SHORt | Set the calibration type for the specified channel to the response calibration (short) of the specified port. |
| SENSe<x>:CORRection:COEFficient:METHod: RESPonse:THRU | Set the calibration type for the specified channel to the same response calibration thru of the specified port (input or output). |

| Command | Description |
|---|---|
| SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS | Set the number of ports used for the SOLT calibration procedure for the specified channel. |
| SENSe<x>;:CORRection:COEFficient:METHod:SOLT:PORTS | Sets the ports to use the SOLT calibration based on the number of ports set by SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS on the specified channel. |
| SENSe<x>:CORRection:COEFficient:METHod:SOLT1 | Set the calibration type a full 1-port calibration for the specified port for the specified channel. |
| SENSe<x>:CORRection:COEFficient:METHod:SOLT2 | Set the calibration type a full 2-port calibration for the specified ports for the specified channel. |
| SENSe<x>:CORRection:COEFficient:SAVE | Calculate and apply the calibration coefficients depending on the calibration type selected when writing the calibration data. When you enable the calibration coefficients, this action clears all calibration data regardless of whether it was used for calculation. This action also clears the calibration type selection. |
| SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation | Measure the port isolation calibration data from the specified stimulus port to the specified response port for the specified channel. |
| SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD | Measure the calibration data of the load standard of the specified port for the specified channel. |
| SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN | Measure the calibration data of the open standard of the specified port for the specified channel. |
| SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt | Measure the calibration data of the short standard of the specified port for the specified channel. |
| SENSe<x>:CORRection:COLLect:[ACQuire]:THRU | Measure the calibration data of the thru standard from the specified stimulus port to the specified response port for the specified channel. |

| Command | Description |
|---|---|
| SENSe<x>:CORRection:COLLect:[ACQuire]:SUBClass | Set or query the subclass used for the calibration for the specified channel. For example, if you set two subclasses thru1 and thru2 (visible in the calibration menu), this command can select either thru1 or thru2 to perform calibration. |
| SENSe<x>:CORRection:COLLect:CKIT:LABel | Set or query the name of the active calibration kit selected for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:ORDer:LABel | Set or query the name of the subclass label selected for the calibration kit for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:ORDer:LOAD | Set or query standard used for the load measurement for the active calibration on the specified port for the specified channel. You must set the specified standard type to load SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE before performing a calibration. |
| SENSe<x>:CORRection:COLLect:CKIT:ORDer:OPEN | Set or query standard used for the open measurement from calibration kit selected for the specified port in the specified channel. You must set the specified standard type to open SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE before performing a calibration. |
| SENSe<x>:CORRection:COLLect:CKIT:ORDer:SHORt | Set or query standard used for the short measurement for the active calibration kit for the specified port at the specified channel, Before performing a calibration, you must set the specified standard type to short by using SENSe<x>:CORRection:COLLect:CKIT: STAN<y>:TYPE. |
| SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect] | Set or query a subclass for calibration from the subclass menu for the specified channel. |

| Command | Description |
|---|---|
| SENSe<x>:CORRection:COLLect:CKIT:ORDer:THRU | Set or query the calibration kit standard selected for the thru measurement between the specified ports for the specified channel. You must set the specified standard type to thru or unknown thru with SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE, before performing a calibration. |
| SENSe<x>:CORRection:COLLect:CKIT:RESet | Resets the specified calibration kit back to the default factory setting state for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:[SELect] | Set or query the index of the calibration kit for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:ARBitrary | Set or query the arbitrary impedance value of the arbitrary standard type of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C0 | Set or query the C0 value for the open standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C1 | Set or query the C1 value for the for the open standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C2 | Set or query the C2 value for the open standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:OPEN for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C3 | Set or query the C3 value for the Open standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:CHARacter | Set or query the type of media for the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:DELay | Set or query the offset delay for the specified calibration kit standard for the specified channel. |

| Command | Description |
|---|---|
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FILE | Set or query the name of the touchstone file for the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMAXimum | Set or query the stop frequency for the specified calibration kit standard for the specified channel. When you select the waveguide type of media, use this command to set the stop frequency of the cutoff frequency. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMINimum | Set or query the start frequency for the specified calibration kit standard for the specified channel. When you select the waveguide type of media, use this command to set the start frequency of the cutoff frequency. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L0 | Set or query the value of L0 for the Short standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt of the specified calibration kit standard (Std) for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L1 | Set or query the value of L1 for the short standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L2 | Set or query the value of L2 for the short standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L3 | Set or query the value of L3 for the short standard type set by SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LABel | Set or query the label of the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LOSS | Set or query the offset loss for the specified calibration kit standard for the specified channel. |

| Command | Description |
|---|---|
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE | Set or query standard type for the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:Z0 | Set or query the characteristic impedance for the specified calibration kit standard for the specified channel. |
| SENSe<x>:CORRection:COLLect:CLEar | Clear calibration measurement data for the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:ERESPonse | Set enhanced response calibration as the calibration method for the specified channel between the stimulus and response ports. |
| SENSe<x>:CORRection:COLLect:METHod:RESPonse:OPEN | Set the calibration type to response calibration open for the specified port and the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt | Set or query the calibration type to response calibration short for the specified port and the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:RESPonse:THRU | Set the calibration type to response calibration thru for the specified ports and the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS | Set the number of ports for SOLT calibration for the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:SOLT:PORTS | Set the ports for SOLT calibration based on the number of ports set by SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS for the specified channel |
| SENSe<x>:CORRection:COLLect:METHod:SOLT1 | Set the calibration type to 1–port calibration for the specified port and the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:SOLT2 | Set the calibration type to full 2–port calibration between the specified ports for the specified channel. |
| SENSe<x>:CORRection:COLLect:METHod:TYPE? | Query the type of calibration for the specified channel. |
| SENSe<x>:CORRection:COLLect:PARTial:SAVE | Use the partial overwrite function and recalculate calibration coefficients depending on the type of calibration and data measured. This action clears all measured data and the selection for calibration type. |

| Command | Description |
|---|---|
| SENSe<x>:CORRection:COLLect:SAVE | Calculate and apply the calibration coefficients based on the type of calibration and data measured. This action clears all measured data and the selection for calibration type. |
| SENSe<x>:CORRection:EXTension:PORT<y>:FREQuency<z> | Set or query the frequency used to calculate the selected loss value for the specified port and channel. |
| SENSe<x>:CORRection:EXTension:PORT<y>:INCLude:[STATe] | Set or query the status of the loss and frequency values for the specified port and channel. |
| SENSe<x>:CORRection:EXTension:PORT<y>:LOSS<z> | Set or query the loss value for loss 1 or 2 at the specified port and channel. |
| SENSe<x>:CORRection:EXTension:PORT<y>:LDC | Set or query the DC loss value for the specified port and channel. |
| SENSe<x>:CORRection:EXTension:PORT<y>:[TIME] | Set or query the delay time for the port extension for the specified port and channel. |
| SENSe<x>:CORRection:EXTension:[STATe] | Set or query the state of the port extension for the specified channel. |
| SENSe<x>:CORRection:RECeiver<y>:COLLect:[ACQuire] | Execute the receiver calibration for the specified port and channel. |
| SENSe<x>:CORRection:RECeiver<y>:[STATe] | Set or query the state of the error correction for the receiver calibration for the specified port and channel. This command only works if the receiver calibration has been previously performed at the specified port. |
| SENSe<x>:CORRection:RVELocity:COAX | Set or query the velocity factory for the specified channel. |
| SENSe<x>:CORRection:[STATe] | Set or query error correction for the specified channel. |
| SENSe<x>:CORRection:TRIGger:FREE | Enable or disable the internal trigger source for calibration in the specified channel. Changing the trigger source during a sweep cancels the sweep. |
| SENSe<x>:CORRection:TYPE? | Query the calibration type used for the selected channel. |
| SENSe<x>:FREQuency:CENTer | Set or query the center frequency value of the sweep range for the specified channel. |
| SENSe<x>:FREQuency:CW | Set or query the CW frequency value of the sweep range of the specified channel. |

| Command | Description |
|---|---|
| SENSe<x>FREQuency:FIXed | Set or query the fixed frequency value used by power sweep for the specified channel. |
| SENSe<x>:FREQuency:DATA? | Query the frequency at all measurement points for the specified channel. |
| SENSe<x>:FREQuency:SPAN | Set or query the span of the sweep range for the specified channel. |
| SENSe<x>:FREQuency:STARt | Set or query the value for the starting frequency of the sweep range for the specified channel. |
| SENSe<x>:FREQuency:STOP | Set or query the value for the stopping frequency of the sweep range for the specified channel. |
| SENSe<x>:SEGMent:DATA | Create or query the array for the segment sweep table on the specified channel. |
| SENSe<x>:SEGMent:SWEep:POINts? | Query the total number of measurement points for the segment sweep for the specified channel. |
| SENSe<x>:SEGMent:SWEep:TIME:[DATA]? | Query the total sweep time (excluding sweep delay) of the segmented sweep for the specified channel. |
| SENSe<x>:SWEep:DELay | Set or query the sweep delay for the specified channel. |
| SENSe<x>:SWEep:POINts | Set or query the number of measurement points for the specified channel. |
| SENSe<x>:SWEep:TIME:AUTO | Set or query the status of the automatic sweep time for the selected channel. |
| SENSe<x>:SWEep:TIME:DATA | Set or query the value of the sweep time for the specified channel. If the specified time value (in seconds) is outside the range, the respective minimum or maximum value is set accordingly. The range varies based on measurement conditions such as IF BW and the number of measurement points. |
| SENSe<x>:SWEep:TYPE | Set or query the sweep type for the selected channel. |
| SENSe<x>:SWEep:GENeration | Set or query the sweep mode of the specified channel. |

# Service commands

Use the SERVice commands to control service operations related to the analyzer.

| Command | Description |
|---|---|
| SERVice:CHANnel:ACTive | Read the number of the active channel. |
| SERVice:CHANnel<x>:COUNt | Display the number of enabled channels. |
| SERVice:CHANnel<x>:TRACe:ACTive? | Display the active trace for the specified channel. |
| SERVice:CHANnel:TRACe:COUNt? | Display the number of traces per channel. |

# Simulator commands

Use SIMulator commands to control the analyzer in simulator (offline) mode.

| Command | Description |
|---|---|
| SIMulator:FILE:FOLDer | Set or query the name of the folder for simulator files. This command does not create a new folder. |
| SIMulator:FILE:FET | Load the CSV file of factory error terms for the specified port and channel. |
| SIMulator:FILE:UET | Load the CSV file containing user error terms for the specified port and channel. |
| SIMulator:FILE:STF | Load the simulator STF CSV file. |
| SIMulator:FILE:PCF | Load the simulator PCF CSV file. |
| SIMulator:FILE:RTF | Load the simulator RTF CSV file. |
| SIMulator:FILE:LCOM | Load the simulator loss compensation file. |
| SIMulator:FILEname | Set or query the touchstone file name that describes the DUT. |
| SIMulator:NF | Set or query the noise floor value for simulator mode. |
| SIMulator:RCF | Set or query the RCF value for the instrument. |

# Source commands

Use SOURce commands to control power calibration.

| Command | Description |
|---|---|
| SOURce<x>:POWer:CENTer | Set or display the power level for the specified channel. |
| SOURce<x>:POWer:[LEVel]:[IMMediate]:[AMPLitude] | Set or query the value for the power level on the specified channel. |
| SOURce<x>:POWer:[LEVel]:SLOPe:[DATA] | Set or query the correction value for the power slope for the specified channel. |
| SOURce<zx>:POWer:[LEVel]:SLOPe:[STATe] | Set or query the status of the power (slope) for the specified channel. |
| SOURce<x>:POWer:PORT<y>:COUPle | Set or query the same power level for all ports of the specified channel. |
| SOURce<x>:POWer:PORT<y>:CORRection:COLLect:[ACQuire] | Set the power calibration data using power sensor A for the specified port and channel. When you complete the calibration measurements, this command turns on the power level error correction. An error occurs if the power sensor is not properly connected. |
| SOURce<xz>:POWer:PORT<y>:CORRection:COLLect:AVERage:[COUNt] | Set or query the averaging factor value for the power calibration of the specified port for the specified channel. |
| SOURce<x>:POWer:PORT<y>:CORRection:COLLect:NTOLerance | Set or query the tolerance value for the power calibration data of the specified port for the specified channel. |
| SOURce<x>:POWer:PORT<y>:CORRection:COLLect:ASENsor:RCFactor | Set or query the value of the reference calibration coefficient for the active power sensor for the specified of in the specified channel. The reference calibration coefficient is the calibration coefficient at 50 MHz. |
| SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:ASENsor:DATA | Set or query the calibration table array for the active power sensor of the specified port at the specified channel. |
| SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:DATA | Set or query the data array representing the loss compensation table for the active power sensor at the specified port of the specified channel. |
| SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:[STATe] | Set or query the loss compensation for the specified port and channel. |

| Command | Description |
|---|---|
| SOURce<x>:POWer:PORT<y>:CORRection:DATA | Set or query the power calibration data array, for the specified port and channel. |
| SOURce<x>:POWer:PORT<y>:CORRection:[STATe] | Set or query the state of the power level error correction for the specified port and channel. |
| SOURce<x>:POWer:PORT<y>:[LEVel]:[IMMediate]:[AMPLitude] | Set or query the value for the power level for the specified port and channel. |
| SOURce<x>:POWer:SPAN | Set or query the span value of the sweep range for the power sweep for the specified channel. |
| SOURce<x>:POWer:STARt | Set or query the starting value of the sweep range for the power sweep on the specified channel. |
| SOURce<x>:POWer:STOP | Set or query the stopping value of the sweep range for the power sweep of the specified channel. |

# Status commands

Use STATus commands to control the status and event registers.

| Command | Description |
|---|---|
| STATus:OPERation:CONDition? | Query the value of the Operation Status Condition Register. |
| STATus:OPERation:ENABle | Set or query the value of the Operation Status Enable Register. |
| STATus:OPERation:[EVENt]? | Query the value of the Operation Status Event Register. |
| STATus:OPERation:NTRansition | Set or query the value of the negative transition filter of the Operation Status Register. |
| STATus:OPERation:PTRansition | Set or query the value of the positive transition filter of the Operation Status Register. |
| STATus:PRESet | Initialize Status Registers. |
| STATus:QUEStionable:BLIMit:CHANnel<x>:CONDition? | Query the value of the Questionable Bandwidth Limit Channel Status Condition Register for the specified channel. The register represents the pass/fail results from the bandwidth limit test for traces 1-14 for the specified channel |
| STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition? | Query the value of the Questionable Ripple Limit Channel Extra Status Condition Register for the specified channel. The register represents the pass/fail results from the bandwidth limit test for traces 15-16 for the specified channel |
| STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:ENABle | Set or query the value of the Questionable Ripple Limit Channel Extra Status Enable Register for the specified channel. This enables the corresponding status event register based on the type of transition used for the respective event bit for traces 15-16. Bits used are 0-1, while bits 2 and 15 are always 0. |
| STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:[EVENt]? | Set or query the value of the Questionable Ripple Limit Channel Extra Status Event Register for the specified channel. For traces 15-16, bits 0-1 use 0 for no event occurred and 1 for an event based on the transition type used for the bandwidth limit test. |

| Command | Description |
|---|---|
| STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition | Set or query the value of the negative transition filter of the Questionable Ripple Limit Channel Extra Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). You must only use traces 15-16 to represent bits 0-1. |
| STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition | Set or query the value of the positive transition filter of the Questionable Ripple Limit Channel Extra Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). You must only use traces 15-16 to represent bits 0-1. |
| STATus:QUEStionable:BLIMit:CHANnel<x>:ENABle | Set or query the value of the Questionable Ripple Limit Channel Status Enable Register for the specified channel. This enables the corresponding status event register, based on the type of transition used for the respective event bit for traces 1-14. Bits used are 1-14, where bits 0 and 15 are always 0. |
| STATus:QUEStionable:BLIMit:CHANnel<x>:[EVENt]? | Set or query the value of the Questionable Ripple Limit Channel Status Event Register for the specified channel. For traces 0-14, bits 0 and 1 use 0 for no event occurred and 1 for an event based on the transition type used for the bandwidth limit test. |
| STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition | Set or query the value of the negative transition filter of the Questionable Ripple Limit Channel Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used to represent bits 1-14 |

| Command | Description |
|---|---|
| STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition | Set or query the value of the positive transition filter of the Questionable Ripple Limit Channel Status Register for the specified channel. You cannot set bits 0 and 3–15 to 1. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used to represent bits 1-14. |
| STATus:QUEStionable:BLIMit:CONDition? | Query the value of the Questionable Ripple Limit Status Condition Register. For channels 1-14, bits 1-14 are defined by 0 for pass and 1 for fail. Bit 0 ORs the enabled bits of the extra status condition register. |
| STATus:QUEStionable:BLIMit:ENABle | Set or query the value of the Questionable Limit Status Enable Register. The preset value for the register varies depending on the upper limit setting for the channel/trace. You cannot set bit 15 of this register to 1. |
| STATus:QUEStionable:BLIMit:[EVENt]? | Query the value of the Questionable Bandwidth Limit Status Event Register. |
| STATus:QUEStionable:LIMit:CHANnel<x>:CONDition? | Query the value of the Questionable Limit Channel Status Condition Register for the specified channel. The register represents the pass/fail results from the limit test for traces 1-14 for the specified channel. |
| STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition? | Query the value of the Questionable Limit Channel Extra Status Condition Register for the specified channel. The register represents the pass/fail result from the limit test for traces 1-14 for the specified channel. |
| STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:ENABle | Set or query the value of the Questionable Limit Channel Extra Status Enable Register for the specified channel. This command enables the corresponding status event register based on the type of transition used for the respective event bit for traces 15-16. Bits used are 0-1, while bits 2 and 15 are always 0. |

| Command | Description |
|---|---|
| STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:[EVENt]? | Set or query the value of the Questionable Limit Channel Extra Status Event Register for the specified channel. For traces 15-16, bits 0-1 indicate 0 for no event occurred and 1 for an event based on the transition type used for limit test. |
| STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition | Set or query the value of value of the negative transition filter of the Questionable Limit Channel Extra Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the limit test, when the test result transitions from fail (1) to pass (0). Traces 15-16 are only used to represent bits 0-1. |
| STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition | Set or query the value of the positive transition filter of the Questionable Limit Channel Extra Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the limit test, when the test result transitions from pass (0) to fail (1). Traces 15-16 are only used to represent bits 0-1. |
| STATus:QUEStionable:LIMit:CHANnel<x>:ENABle | Set or query the value of the Questionable Limit Channel Status Enable Register for the specified channel. This enables the corresponding status event register based on the type of transition used for the respective event bit for traces 1-14. Bits used are 1-14, where bits 0 and 15 are always 0. |
| STATus:QUEStionable:LIMit:CHANnel<x>:[EVENt]? | Query the value of the Questionable Limit Channel Status Event Register for the specified channel. |
| STATus:QUEStionable:LIMit:CHANnel<x>:NTRansition | Set or query the value of the negative transition filter of the Questionable Limit Channel Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the limit test, when the test result transitions from fail (1) to pass (0). Traces 1-14 are only used to represent bits 1-14. |

| Command | Description |
|---|---|
| STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition | Set or query the value of the positive transition filter of the Questionable Limit Channel Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used to represent bits 1-14. |
| STATus:QUEStionable:LIMit:CONDition? | Query the value of the Questionable Limit Channel Status Condition Register. |
| STATus:QUEStionable:LIMit:ELIMit:CONDition? | Query the value of the Questionable Limit Extra Status Condition Register. |
| STATus:QUEStionable:LIMit:ELIMit:ENABle | Set or query the value of the Questionable Limit Extra Status Enable Register. |
| STATus:QUEStionable:LIMit:ELIMit:EVENt? | Query the value of the Questionable Limit Extra Status Event Register. |
| STATus:QUEStionable:LIMit:ELIMit:NTRansition | Set or query the value of value of the negative transition filter of the Questionable Limit Extra Status Register. You cannot set bits 0 and 3–15 to 1. |
| STATus:QUEStionable:LIMit:ELIMit:PTRansition | Set or query the value of the positive transition filter of the Questionable Limit Extra Status Register. You cannot set bits 0 and 3–15 to 1. |
| STATus:QUEStionable:LIMit:ENABle | Set or query the value of the Questionable Limit Status Enable Register. |
| STATus:QUEStionable:LIMit:[EVENt]? | Query the value of the Questionable Limit Status Event Register. |
| STATus:QUEStionable:LIMit:NTRansition | Set or query the value of value of the negative transition filter of the Questionable Limit Status Register. You cannot set bits 0 and 3–15 to 1. |
| STATus:QUEStionable:LIMit:PTRansition | Set or query the value of the positive transition filter of the Questionable Limit Status Register for the specified channel. You cannot set bits 0 and 3–15 to 1. |
| STATus:QUEStionable:NTRansition | Set or query the value of the negative transition filter of the Questionable Status Register. You cannot set bits 0 and 3–15 to 1. |
| STATus:QUEStionable:PTRansition | Set or query the value of the positive transition filter of the Questionable Status Register for the specified channel. You cannot set bits 0 and 3–15 to 1. |

| Command | Description |
|---|---|
| STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition? | Query the value of the Questionable Ripple Limit Channel Status Condition Register for the specified channel. The register represents the pass/fail results from the ripple limit test for traces 1-14 on the specified channel. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:CONDition? | Query the value of the Questionable Ripple Limit Channel Extra Status Condition Register for the specified channel. The register represents the pass/fail results from the ripple limit test results for traces 15-16 on the specified channel. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:ENABle | Set or query the value of the Questionable Ripple Limit Channel Extra Status Enable Register for the specified channel. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:[EVENt]? | Query the value of the Questionable Ripple Limit Channel Extra Status Event Register for the specified channel. For traces 15-16, the bits 0-1 indicate 0 for no event occurred and 1 for an event based on the transition type used for the ripple limit test. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:NTRansition | Set or query the value of the negative transition filter of the Questionable Ripple Limit Channel Extra Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the ripple limit test, when the test result transitions from fail (1) to pass (0). Traces 15-16 are only used, representing bits 0-1. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:PTRansition | Set or query the value of the positive transition filter of the Questionable Ripple Limit Channel Extra Status Register for the specified channel. A positive transition is defined when the condition register changes from a 0 to a 1. This occurs during the ripple limit test, when the test result transitions from pass (0) to fail (1). Traces 15-16 are only used, representing bits 0-1. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:ENABle | Set or query the value of the Questionable Ripple Limit Channel Status Enable Register for the specified channel. |

| Command | Description |
|---|---|
| STATus:QUEStionable:RLIMit:CHANnel<x>:[EVENt]? | Query the value of the Questionable Ripple Limit Channel Status Event Register for the specified channel. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:NTRansition | Set or query the value of the negative transition filter of the Questionable Ripple Limit Channel Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the ripple limit test, when the test result transitions from fail (1) to pass (0). Traces 1-14 are only used, representing bits 1-14. |
| STATus:QUEStionable:RLIMit:CHANnel<x>:PTRansition | Set or query the value of the positive transition filter of the Questionable Ripple Limit Channel Status Register for the specified channel. A positive transition is defined when the condition register changes from a 0 to a 1. This occurs during the ripple limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used, representing bits 1-14. |
| STATus:QUEStionable:RLIMit:CONDition? | Query the value of the Questionable Ripple Limit Status Condition Register. |
| STATus:QUEStionable:RLIMit:ELIMit:CONDition? | Query the value of the Questionable Ripple Limit Extra Status Condition Register. |
| STATus:QUEStionable:RLIMit:ENABle | Set or query the value of the Questionable Ripple Limit Status Enable Register. |
| STATus:QUEStionable:RLIMit:[EVENt]? | Set or query the value of the Questionable Ripple Limit Status Event Register. |
| STATus:QUEStionable:RLIMit:NTRansition | Set or query the value of the negative transition filter of the Questionable Ripple Limit Status Register. You cannot set bits 0 and 3–15 to 1. |
| STATus:QUEStionable:RLIMit:PTRansition | Set or query the value of the positive transition filter of the Questionable Ripple Limit Status Register. You cannot set bits 0 and 3–15 to 1. |

# System commands

Use SYSTem commands to control beep settings, warnings, and errors.

| Command | Description |
|---|---|
| SYSTem:BEEPer:WARNing:IMMediate | Generate an immediate beep to verify if the warning beep works properly. |
| SYSTem:BEEPer:WARNing:[STATe] | Set or query the status of the beeper, which indicates a warning, error or an out of range limit. |
| SYSTem:BIAStee | Set or query the state of the bias tee function. |
| SYSTem:COMMunicate:VISA:PMETer:MODel | Set or display the model name of the power meter used with the TTR500 instrument. |
| SYSTem:COMMunicate:VISA:PMETer:ADDRess | Set or query the VISA address of the power meter used with the TTR500 instrument. with the TTR500 instrument. |
| SYSTem:CONNect | Set or query the VISA address of the power meter used with the TTR500 instrument. |
| SYSTem:CORRection:[STATe] | Set or query the status of system error correction. |
| SYSTem:CORRection:PERFormance | Set or query the status of system error correction. |
| SYSTem:DISCOVER? | Query the list of disaggregate instruments available on the USB bus along with the simulator. |
| SYSTem:DISConnect | Query the list of disaggregate instruments available on the USB bus along with the simulator. |
| SYSTem:ERRor:ALL? | Display information about all errors and events. |
| SYSTem:ERRor:CODE:ALL? | Display information about all errors and events. |
| SYSTem:ERRor:CODE:[NEXT]? | Query the code numbers for all errors and events. |
| SYSTem:ERRor:COUNt? | Display the total number of errors and events. |
| SYSTem:ERRor:[NEXT]? | Query the error code, description and details about the next error or event code. |
| SYSTem:PRESet:MODE | Set or query the preset mode for the TTR500 instrument. |
| SYSTem:SERVice? | Query the operating mode (normal or service mode) for the TTR500 instrument. |

| Command | Description |
|---|---|
| SYSTem:UPReset:FILEname | Set the user preset mode for the TTR500 instrument using a file.. |
| SYSTem:TEMPerature:[STATe]? | Query the warm up state of the TTR500 instrument. |
| SYSTem:STACking:TEST | Set or query the state of the stacking test for the TTR500 instrument. |
| SYSTem:STACking:TIMing | Set or query the state of the stacking timing test for the TTR500 instrument. |
| SYSTem:TEMPerature:DATA? | Query the temperature sensor data for the TTR500 instrument. |
| SYSTem:TIMing:DATA | Set or query the stacking timing test data for the TTR500 instrument. |

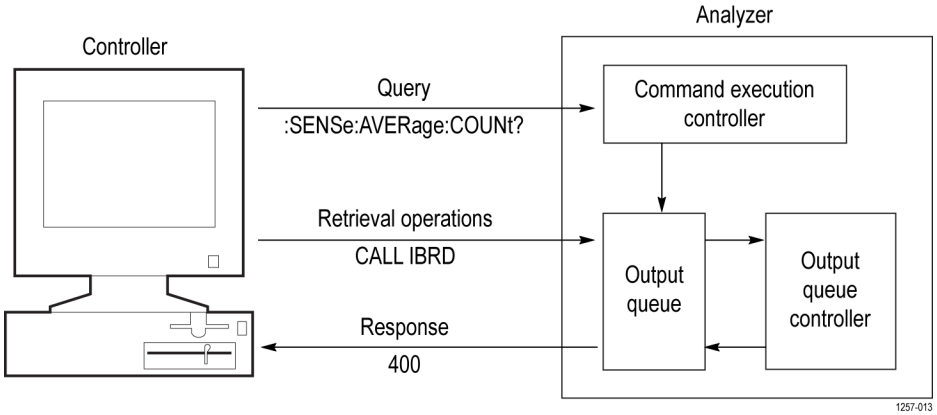
# Trigger commands

Use TRIGger commands to control the triggering function.

| Command | Description |
|---|---|
| TRIGger:AVERage | Set or query the averaging trigger function. |
| TRIGger:EXTernal:DELay | Set or query the external trigger delay. This is the time taken by the VNA to start a measurement upon receipt of trigger when the source is external. |
| TRIGger:[SEQuence]:[IMMediate] | Generate a trigger sequence immediately, regardless of the settings of the SCPI manual trigger. This command generates a trigger and executes a measurement, regardless of the settings of the SCPI manual trigger mode. Unlike TRIGger:SEQuence:SINGle, the execution of the object finishes at the time of a trigger. If you execute this object when the trigger system is not in the trigger wait and armed state, an error occurs. |
| TRIGger:[SEQuence]:POINt | Set or query the point trigger function. |
| TRIGger:[SEQuence]:SINGle | Generate a trigger immediately and execute a measurement, regardless of the settings of the SCPI manual trigger mode. Unlike TRIGger:SEQuence:IMMediate, the execution of the object finishes when the measurement (of all of the sweep) initiated with this object is complete. You can wait for the end of the measurement using the OPC object. If you execute this object when the trigger system is not in the trigger wait state, an error occurs. |
| TRIGger:[SEQuence]:SCOPe | Use triggers to synchronize measurements with external events. The trigger function requires VectorVu-PC to have an active connection to a TTR500 unit. The trigger function does not work in simulator mode. |
| TRIGger:[SEQuence]:SOURce | Set or query the source for the trigger. |

# Retrieving Response Message

When receiving a query command from the external controller, the analyzer puts the response message on the Output Queue. This message cannot be retrieved unless you perform retrieval operations to the external controller. (For example, call the IBRD subroutine included in the GPIB software of National Instruments.)




When the Output Queue contains a response message, sending another command from the external controller before retrieving this message deletes it from the queue. The Output Queue always contains the response message to the most recent query command.

You can use the MAV bit of the Status Byte Register (SBR) to check whether the Output Queue contains a response message. For details, refer to *Status Byte Register (SBR)*.

# Command Descriptions

## ABORt (No Query Form)

Abort the current measurement and change the trigger sequence to idle state for all channels.

**Group** Abort commands

**Syntax**

```
ABORt
```

**Examples** `ABORT` stops the current measurement and sets trigger to idle state.

## CALCulate<x>:PARameter:COUNt

Set or query the number of traces on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:PARameter:COUNt <value>
CALCulate<x>:PARameter:COUNt?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of traces displayed.

**Returns** `<NR1>`

**Examples** `CALC:PAR:COUNt 1` sets 1 trace to show on channel 1 (default).

`CALC2:PAR:COUNt?` may return 3, as the number of traces displayed on channel 2.

`CALC3:PAR:COUNt 2` sets 2 traces to show on channel 3.

## CALCulate<x>:PARameter<y>:DEFine

Set or query a measurement parameter of the specified trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:PARameter<y>:DEFine <string>
CALCulate<x>:PARameter<y>:DEFine?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<string>` is S11, S12, S21, S22, A, B, R1, R2.

**Returns** `<string>`

**Examples** `CALC1:PAR5:DEF S11` calculates the S11 parameter of trace 5 on channel 1.

`CALC1:PAR5:DEF?` may return S22, which is the measurement parameter set for trace 5 on channel 1.

## CALCulate<x>:PARameter<y>:RPORt

Set or query the receiver port of the selected trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:PARameter<y>:RPORt <value>
CALCulate<x>:PARameter<y>:RPORt?
```

**Related Commands** CALCulate<x>:PARameter<y>:SPORt

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<NR1>` is the receiver port (1 or 2).

**Returns** `1 or 2`

**Examples** `CALC1:PAR3:RPOR 2` sets the receiver port 2 of trace 3 on channel 1.

`CALC3:PAR1:RPOR?` may return 1, which is the receiver port number of trace 1 on channel 3.

## CALCulate<x>:PARameter<y>:[SELect] (No Query Form)

Set the selected trace as the active trace on the specified channel.

**Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:PARameter<y>:[SELect]
```

**Arguments** None

**Examples** `CALC2:PAR4:SEL` sets trace 4 as the active trace on channel 2.

`CALC3:PAR3` sets trace 3 not to be the active trace on channel 3.

## CALCulate<x>:PARameter<y>:SPORt

Set or query the source port of the specified trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:PARameter<y>:SPORt <value>
CALCulate<x>:PARameter<y>:SPORt?
```

**Related Commands** CALCulate<x>:PARameter<y>:RPORt

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<NR1>` is the source port (1 or 2).

**Returns** `1 or 2`

**Examples** `CALC4:PAR2:SPOR 1` sets source port 1 of trace 2 on channel 4.

`CALC1:PAR3:SPOR?` may return 2, which is the source port number of trace 3 on channel 1.

## CALCulate<x>:[SELected]:BLIMit:DB

Set or query the bandwidth threshold value (dB) for the bandwidth test of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:DB <value>
CALCulate<x>:[SELected]:BLIMit:DB?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the bandwidth. Range: 0 to 200 dB.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:BLIM:DB 3` sets the bandwidth at 3 dB for channel 1.

`CALC1:SEL:BLIM:DB?` may return 50, which means 50 dB is the bandwidth threshold value of the bandwidth test set for channel 1.

## CALCulate<x>:[SELected]:BLIMit:DISPlay:MARKer

Set or query the state of the display for the marker of the bandwidth test for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:DISPlay:MARKer <value>
CALCulate<x>:[SELected]:BLIMit:DISPlay:MARKer?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the marker for the bandwidth test shows.

0 means the marker for the bandwidth test is hidden.

**Examples** `CALC1:SEL:BLIM:DISP:MARK 1` enables the display of the marker for the bandwidth test for the active trace on channel 1.

`CALC4:SEL:BLIM:DISP:MARK?` may return 0, which means the display of the marker for the bandwidth test is disabled for the active trace on channel 4.

## CALCulate<x>:[SELected]:BLIMit:DISPlay:VALue

Set or query the state of the display for the bandwidth value of the bandwidth test of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:DISPlay:VALue <value>
CALCulate<x>:[SELected]:BLIMit:DISPlay:VALue?
```

**Related Commands** CALCulate<x>:[SELected]:BLIMit:DISPlay:MARKer

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the bandwidth value shows.

0 means the bandwidth value is hidden.

**Examples** `CALC1:SEL:BLIM:DISP:VAL 1` enables the display of the bandwidth value for the bandwidth test for the active trace on channel 1.

`CALC2:SEL:BLIM:DISP:VAL?` may return 0, which means the bandwidth value for the bandwidth test is disabled for the active trace on channel 2.

## CALCulate<x>:[SELected]:BLIMit:FAIL? (Query Only)

Query the result for the bandwidth test of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:FAIL?
```

**Related Commands** CALCulate<x>:[SELected]:BLIMit:[STATe]

**Returns** **Command variables.** <x> is the channel number (1 to 16).

1 means the bandwidth test failed.

0 means the bandwidth test passed.

**Examples** `CALC1:SEL:BLIM:FAIL?` may return 1, which means the bandwidth test failed for the active trace on channel 1.

`CALC1:SEL:BLIM:FAIL?` may display 0, which means the bandwidth test passed for the active trace on channel 1.

## CALCulate<x>:[SELected]:BLIMit:MAXimum

Set or query the upper limit value for the bandwidth test of the active trace on the specified channel. When the upper limit value is set to power sweep, the maximum and minimum values go outside the dB range.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:MAXimum <value>
CALCulate<x>:[SELected]:BLIMit:MAXimum?
```

**Related Commands** CALCulate<x>:[SELected]:BLIMit:MINimum

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the upper limit value of the bandwidth test in Hz. Range: 0 to 6e9.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:BLIM:MAX 1e9` sets the upper limit value of the bandwidth test to 1 GHz for the active trace on channel 1.

`CALC3:SEL:BLIM:MAX?` may return 2e9, which means 2 GHz is the upper limit value of the bandwidth test set for the active trace on channel 3.

## CALCulate<x>:[SELected]:BLIMit:MINimum

Set or query the lower limit value (in Hz) for the bandwidth test in the active trace for the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:MINimum <value>
CALCulate<x>:[SELected]:BLIMit:MINimum?
```

**Related Commands** CALCulate<x>:[SELected]:BLIMit:MAXimum

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the lower limit of value for the bandwidth test. Range: 0 to 6e9 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:BLIM:MIN 1e6` sets the lower limit value to 1 MHz for the bandwidth test for the active trace on channel 1.

`CALC3:SEL:BLIM:MIN?` may return 4e9, which means 4 GHz is the lower limit value for the bandwidth test set for the active trace on channel 3.

## CALCulate<x>:[SELected]:BLIMit:REPort:DATA? (Query Only)

Query the bandwidth value for the bandwidth test of the active trace on the specified channel. The bandwidth test must be enabled using `CALCulate<x>:[SELected]:BLIMit:[STATe]` in order to properly use this command. The bandwidth test returns a value in power sweep.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:REPort:DATA?
```

**Related Commands** CALCulate<x>:[SELected]:BLIMit:[STATe]

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NRf>`

Range: 0 to 6e9 mdB

**Examples** `CALC1:SEL:BLIM:REP:DATA?` may return 5e8, which means 0.5 GHz is the bandwidth value for the bandwidth test set for the active trace on channel 1.

## CALCulate<x>:[SELected]:BLIMit:[STATe]

Set or query the bandwidth limit test for the active trace for the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:BLIMit:[STATe] <value>
CALCulate<x>:[SELected]:BLIMit:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 means the bandwidth limit test is disabled.

1 means the bandwidth limit test is enabled.

**Examples** `CALC1:SEL:BLIM:STAT 1` enables the bandwidth limit test for the active trace on channel 1.

`CALC3:SEL:BLIM:STAT?` may return 0, which means the bandwidth limit test is disabled for the active trace on channel 3.

## CALCulate<x>:[SELected]:CORRection:EDELay:MEDium

Set or query the medium for the electrical delay on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:CORRection:EDELay:MEDium { COAX |
WAV }
CALCulate<x>:[SELected]:CORRection:EDELay:MEDium?
```

**Related Commands** CALCulate<x>:[SELected]:CORRection:EDELay:TIME

CALCulate<x>:[SELected]:CORRection:EDELay:WGCutoff

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`COAXial` sets a coaxial medium for the electrical delay.

`WAVeguide` sets a waveguide medium for the electrical delay.

**Returns** `COAXial` means a coaxial medium is set for the electrical delay.

`WAVeguide` means a waveguide medium is set for the electrical delay.

**Examples** `CALC1:SEL:CORR:EDEL:MED WAV` sets a waveguide medium for the electrical delay on channel 1.

`CALC2:SEL:CORR:EDEL:MED?` may return COAX, which means the coaxial the medium is set for the electrical delay on channel 2.

## CALCulate<x>:[SELected]:CORRection:EDELay:TIME

Set or query the time for the electrical delay for the active trace on the specified channel. If the specified delay time is outside the range, the minimum or maximum time is set accordingly.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:CORRection:EDELay:TIME <value>
CALCulate<x>:[SELected]:CORRection:EDELay:TIME?
```

**Related Commands** CALCulate<x>:[SELected]:CORRection:EDELay:MEDium

CALCulate<x>:[SELected]:CORRection:EDELay:WGCutoff

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the time measured in seconds for the electrical delay. Range: -10 to 10 s.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:CORR:EDEL:TIME 5` sets the electrical delay to 5 seconds for the active trace on channel 1.

`CALC4:SEL:CORR:EDEL:TIME?` may return -3, which means -3 seconds is the electrical delay set for the active trace on channel 4.

## CALCulate<x>:[SELected]:CORRection:EDELay:WGCutoff

Set or query the cutoff frequency (in Hz) when a waveguide delay is used for the electrical delay on the specified channel. If the specified frequency is outside the range, the minimum or maximum frequency is set accordingly.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:CORRection:EDELay:WGCutoff <value>
CALCulate<x>:[SELected]:CORRection:EDELay:WGCutoff?
```

**Related Commands** CALCulate<x>:[SELected]:CORRection:EDELay:MEDium

CALCulate<x>:[SELected]:CORRection:EDELay:TIME

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the cutoff frequency for the selected channel. Range: 1e5 to 6e9 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:CORR:EDEL:WGC 4e5` sets the waveguide cutoff frequency to 400 kHz on channel 1.

`CALC2:SEL:CORR:EDEL:WGC?` may return 2e5, which means 200 kHz is the cutoff frequency set for channel 2.

## CALCulate<x>:[SELected]:CORRection:OFFSet:PHASe

Set or query the phase offset of the active trace on the specified channel. If the specified phase offset is outside the range, the minimum or maximum offset is set accordingly.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:CORRection:OFFSet:PHASe <value>
CALCulate<x>:[SELected]:CORRection:OFFSet:PHASe?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the phase offset for the active trace for the specified channel. Range: -360 to 360 degrees.

**Returns** `<NR1>`

**Examples** `CALC1:SEL:CORR:OFFS:PHAS 45` sets the phase offset at 45 degrees for the active trace on channel 1.

`CALC3:SEL:CORR:OFFS:PHAS?` may return -30, which means -30 degrees is the phase offset set for the active trace on channel 3.

## CALCulate<x>:[SELected]:DATA:FDATa

Set or query the formatted data array of the active trace on the specified channel. To set this command, you must provide two times the number of points as the number of points on the active trace. This is to account for both main and auxiliary values.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:DATA:FDATa <value>
CALCulate<x>:[SELected]:DATA:FDATa?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the formatted data array of NP*2 for any value *n* between 1 and NP, where:

`Data(n*2-1)` is the main response value.

`Data(n*2)` is the auxiliary response value.

***NOTE.*** *Rectangular display formats return 0 (zero) value, which means there is no auxiliary response value for this display format type. Displays in Smith Chart and Polar formats, however, contain an auxiliary response value.*

**Returns** `[numeric1],[numeric2],[numeric3],...[numeric600]`

**Examples** `CALC1:SEL:DATA:FDATa?` may return 1.52, 0,0.905 , 0, if the active trace on channel 1 has only two measurement points and the display format is set to log magnitude.

## CALCulate<x>:[SELected]:DATA:FMEMory? (Query Only)

Query the formatted memory array of the active trace of on specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:DATA:FMEMory?
```

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`[numeric1],[numeric2],[numeric3],...[numeric600]` is the formatted memory data array of NP*2 for any value *n* between 1 and NP, where:

`Data(n*2-1)` is the data primary value at the $n^{th}$ measurement point.

`Data(n*2)` is the data secondary value at the $n^{th}$ measurement point.

**Examples** `CALC1:SEL:DATA:FMEM?` may return 0.99258, -0.0019685, 0.17248, -0.984893, which is the formatted memory array of the active trace on channel 1 has only 2 measurement points and the display format is set to Smith Chart format.

## CALCulate<x>:[SELected]:DATA:RDATa

Set or query the raw data array for the active trace on the specified channel. To set this command, you must set the same number of points as the number of points contained on the active trace.

***NOTE.*** *A rectangular display format returns a zero value, which means there is no auxiliary response value for it. Displays in the Smith Chart and Polar formats, however, contain an auxiliary response value.*

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:DATA:RDATa <value>
CALCulate<x>:[SELected]:DATA:RDATa?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the raw data array of NP*2 for any value *n* between 1 and NP, where:

`Data(n*2-1)` is the data real value at the$n^{th}$ measurement point.

`Data(n*2)` is the data imaginary value at the *n th* measurement point.

**Returns** `[numeric1],[numeric2],[numeric3],...[numeric`*n*`]`

**Examples** `CALC1:SEL:DATA:RDAT?` may return 0.990635, -0.0150547, 0.0531267, 1.17447, which is the raw data array if the active trace on channel 1 has only 2 measurement points.

## CALCulate<x>:[SELected]:DATA:SDATa

Set or query the corrected data array of the active trace on the specified channel. To set this command, you must provide two times the number of points as the active trace.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:DATA:SDATa <value>
CALCulate<x>:[SELected]:DATA:SDATa?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the corrected data array of NP*2 for any value *n* between 1 and NP, where:

`Data(n*2-1)` is the data real value at the *n th* measurement point.

`Data(n*2)` is the data imaginary value at the *n th* measurement point.

**Returns** `[numeric1],[numeric2],[numeric3],...[numeric600]`

**Examples** `CALC1:SEL:DATA:SDAT?` may return 1.52, -0.3301, 0.905, -0.77517, which is the corrected data array of the active trace on channel 1 has only 2 measurement points and the display format is set to log magnitude.

## CALCulate<x>:[SELected]:DATA:SMEMory? (Query Only)

Query the corrected memory array for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:DATA:SMEMory?
```

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

`[numeric1],[numeric2],[numeric3],...[numeric`*n*`]` is the corrected memory array of `NP*2` for any value *n* between 1 and NP, where:

`Data(n*2-1)` is the data (real) value at the *n th* measurement point.

`Data(n*2)` is the data (imaginary) value at the *n th* measurement point.

**Examples**

`CALC1:SEL:DATA:SMEM?` may return 0.992031, -0.0011153, 0.171845, -0.986575, which is the corrected memory array for the active trace on channel 1 has only 2 measurement points.

## CALCulate<x>:[SELected]:FORMat

Set or query the data format of the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FORMat <string>
CALCulate<x>:[SELected]:FORMat?
```

**Related Commands**

CALCulate<x>:[SELected]:PHASe

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<string>` is the data format of the active trace on the specified channel. Select one of these formats:

Select one of the following arguments:

| Argument | Description |
|---|---|
| MLOGarithmic | Log magnitude |
| PHASe | Phase |
| GDELay | Group delay |
| SLINear | Smith chart (Linear/Phase) |
| SLOGarithmic | Smith chart (Log/Phase) |
| SCOMplex | Smith chart (Real/Imaginary) |
| SMITh | Smith chart (R+jX) |
| SADMittance | Smith chart (G+jB) |

| Argument | Description |
|---|---|
| PLINear | Polar (Linear/Phase) |
| PLOGarithmic | Polar (Log/Phase) |
| POLar | Polar (Real/Imaginary) |
| MLINear | Linear magnitude |
| SWR | Standing Wave Ratio |
| REAL | Real |
| IMAGinary | Imaginary |
| UPHase | Expanded phase |
| PPHase | Positive phase |

**Returns** See Arguments.

**Examples** `CALC1:SEL:FORM SMIT` sets the Smith Chart data format for the active trace on channel 1.

`CALC5:SEL:FORM?` may return MLOG, which means the data format set for the active trace on channel 5 is Log Magnitude.

## CALCulate<x>:[SELected]:FUNCtion:DATA? (Query Only)

The query returns an alphanumeric value for the analysis results obtained from `CALCulate<x>:[SELected]:FUNCtion:EXECute` based on the analysis set by `CALCulate<x>:[SELected]:FUNCtion:TYPE` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:FUNCtion:DATA?`

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:EXECute

CALCulate<x>:[SELected]:FUNCtion:TYPE

CALCulate<x>:[SELected]:FUNCtion:POINts?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`[numeric1],[numeric2],[numeric3],..[numericn*2]` is the data array of `NP*2.`

Query the number of data points using `CALCulate<x>:[SELected]:FUNCtion:POINts?`. For any value *n* between 1 and N:

`Data(n*2-1)` is the response value or analysis result of the searched *n* *th* measurement point.

`Data(n*2)` is the stimulus value of the*n* *th* measurement point. This value is always 0 for the following analysis types: mean value, standard deviation, and the difference between minimum and maximum values.

**Examples** `CALC1:SEL:FUNC:DATA?` may return 0.119004, 0, which is the analysis results for peak to peak set for the active trace on channel 1.

## CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle

Set or query the analysis range to all traces on the specified channel. The query returns whether the analysis range to all traces on the specified channel is decoupled (0) or coupled (1).

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle <value>
CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 means the analysis range is decoupled.

1 means the analysis range is coupled.

**Examples** `CALC1:SEL:FUNC:DOM:COUP 1` couples the analysis range to all traces on channel 1.

`CALC2:SEL:FUNC:DOM:COUP?` may return 0, which means the analysis range is decoupled to all traces on channel 2.

## CALCulate<x>:[SELected]:FUNCtion:DOMain:STARt

Set or query the starting stimulus value for the arbitrary analysis range (in Hz) used by `CALCulate<x>:SELected]:FUNCtion:EXECute` for the specified

channel. If the analysis range is coupled, the range is set for all traces on the specified channel, otherwise, the active trace start value is set instead.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:DOMain:STARt <value>
CALCulate<x>:[SELected]:FUNCtion:DOMain:STARt?
```

**Related Commands**

CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle

CALCulate<x>:[SELected]:FUNCtion:DOMain:[STATe]

CALCulate<x>:[SELected]:FUNCtion:DOMain:STOP

CALCulate<x>:[SELected]:FUNCtion:EXECute

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the starting value of the arbitrary analysis. Range: 1e5 to 6e9 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:FUNC:DOM:STAR 1e9` sets the starting stimulus value of the arbitrary analysis range at 1 GHz for channel 1.

`CALC2:SEL:FUNC:DOM:STAR?` may return 4e9, which means 4 GHz is the starting stimulus value set for the arbitrary analysis range for channel 2.

## CALCulate<x>:[SELected]:FUNCtion:DOMain:[STATe]

Set or query whether to use an arbitrary range or an entire sweep range on the specified channel, when executing the analysis with `CALCulate<x>:[SELected]:FUNCtion:EXECute`.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:DOMain:[STATe] <value>
CALCulate<x>:[SELected]:FUNCtion:DOMain:[STATe]?
```

**Related Commands**

CALCulate<x>:[SELected]:FUNCtion:EXECute

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP

CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means an arbitrary range is set.

0 means an entire sweep range is set.

**Examples** `CALC1:SEL:FUNC:DOM:STAT 1` sets an arbitrary range for channel 1.

`CALC2:SEL:FUNC:DOM:STAT?` may return 0, which means an entire sweep range is set for channel 2.

## CALCulate<x>:[SELected]:FUNCtion:DOMain:STOP

Set or query the stopping stimulus value (in Hz) for the arbitrary analysis range used by `CALCulate<x>:[SELected]:FUNCtion:EXECute` on the specified channel. If the analysis range is coupled, the range is set for all traces on the specified channel. Otherwise, the active trace stopping value is set instead.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:DOMain:STOP <value>
CALCulate<x>:[SELected]:FUNCtion:DOMain:STOP?
```

**Related Commands**

CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle

CALCulate<x>:[SELected]:FUNCtion:DOMain:STARt

CALCulate<x>:[SELected]:FUNCtion:DOMain:[STATe]

CALCulate<x>:[SELected]:FUNCtion:DOMain:COUPle

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the stopping stimulus value for the arbitrary analysis range. Range: 1e5 to 6e9 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:FUNC:DOM:STOP 1e9` sets the stopping stimulus value for the arbitrary analysis range to 1 GHz for channel 1.

`CALC2:SEL:FUNC:DOM:STOP?` may return 5e9, which means 5 GHz is the stopping stimulus value set for the arbitrary analysis range for channel 2.

## CALCulate<x>:[SELected]:FUNCtion:EXECute (No Query Form)

Execute the statistical analysis set by `CALCulate<x>:[SELected]:FUNCtion:TYPE` for the active trace on the specified channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:EXECute
```

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:DATA?

CALCulate<x>:[SELected]:FUNCtion:DOMain:[STATe]

CALCulate<x>:[SELected]:FUNCtion:TYPE

**Arguments** None

**Examples** `CALC1:SEL:FUNC:EXEC` executes the statistical analysis for the active trace on channel 1.

## CALCulate<x>:[SELected]:FUNCtion:PEXCursion

Set or query the lower limit (in Hz) of the peak excursion value when performing a peak search using `CALCulate<x>:[SELected]:FUNCtion:EXECute` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:PEXCursion <value>
CALCulate<x>:[SELected]:FUNCtion:PEXCursion?
```

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:PPOLarity

CALCulate<x>:[SELected]:FUNCtion:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the lower limit of the peak excursion value. Range: 0 to 5e8 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:FUNC:PEXC 1e8` sets the lower limit of the peak excursion value for the active trace on channel 1.

`CALC2:SEL:FUNC:PEXC?` may return 2e8, which means 0.2 GHz is the lower limit of the peak excursion value set for the active trace on channel 2.

## CALCulate<x>:[SELected]:FUNCtion:POINts? (Query Only)

Queries the number of data pairs found on the analysis result when using `CALCulate<x>:[SELected]:FUNCtion:EXECute` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:FUNCtion:POINts?`

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:DATA?

CALCulate<x>:[SELected]:FUNCtion:EXECute

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the number of data pairs found on the analysis result.

**Examples** `CALC1:SEL:FUNC:POIN?` may return 490, which is the number of data pairs found for the active trace on channel 1.

## CALCulate<x>:[SELected]:FUNCtion:PPOLarity

Set or query the polarity of the peak excursion search when performing a peak search using `CALCulate<x>:[SELected]:FUNCtion:EXECute` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:PPOLarity { POS | NEG |
BOTH }
CALCulate<x>:[SELected]:FUNCtion:PPOLarity?
```

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:PEXCursion

CALCulate<x>:[SELected]:FUNCtion:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`POSitive` is a positive peak excursion search.

`NEGative` is a negative peak excursion search.

`BOTH` is a peak excursion search for positive and negative peaks.

**Returns** `POSitive` means the polarity of the peak excursion search is positive.

`NEGative` means the polarity of the peak excursion search is negative.

`BOTH` means the polarity of the peak excursion search is positive and negative.

**Examples** `CALC1:SEL:FUNC:PPOL POS` sets the positive polarity of the peak search for the active trace on channel 1.

`CALC3:SEL:FUNC:PPOL?` may return BOTH, to indicate a positive and negative polarity are set for the peak search for the active trace on channel 3.

## CALCulate<x>:[SELected]:FUNCtion:TARGet

Set or query a target value (in Hz) for the target search when using `CALCulate<x>:[SELected]:FUNCtion:EXECute` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:TARGet <value>
CALCulate<x>:[SELected]:FUNCtion:TARGet?
```

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:TTRansition

CALCulate<x>:[SELected]:FUNCtion:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the target value for the target search. Range: -5e8 to 5e8 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:FUNC:TARG 3e8` sets the target value at 300 MHz for the target search for the active trace on channel 1.

`CALC2:SEL:FUNC:TARG?` may return -2e8, which means -200 MHz is the target value set for the target search for the active trace on channel 2.

## CALCulate<x>:[SELected]:FUNCtion:TTRansition

Set or query the transition type of the target search when using `CALCulate<x>:[SELected]:FUNCtion:EXECute` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:TTRansition { POS | NEG |
BOTH }
CALCulate<x>:[SELected]:FUNCtion:TTRansition?
```

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:TARGet

CALCulate<x>:[SELected]:FUNCtion:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`POSitive` is the positive target transition.

`NEGative` is the negative target transition.

`BOTH` is the target transition for positive and negative peaks.

**Returns** `POSitive` means the target transition is positive.

`NEGative` means the target transition is negative.

`BOTH` means it is the target transition for positive and negative peaks.

**Examples**

`CALC1:SEL:FUNC:TTR POS` sets the transition type to positive for the target search for the active trace on channel 1.

`CALC2:SEL:FUNC:TTR?` may return NEGative, which means a negative transition is set for the target search for the active trace on channel 2.

## CALCulate<x>:[SELected]:FUNCtion:TYPE

Set or query the analysis type when using `CALCulate<x>:[SELected]:FUNCtion:EXECute` for the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:FUNCtion:TYPE { PTPeak | STDEV |
MEAN | MAXimum | MINimum | PEAK | APEak | ATARget }
CALCulate<x>:[SELected]:FUNCtion:TYPE?
```

**Related Commands**

CALCulate<x>:[SELected]:FUNCtion:TARGet

CALCulate<x>:[SELected]:FUNCtion:TTRansition

CALCulate<x>:[SELected]:FUNCtion:PPOLarity

CALCulate<x>:[SELected]:FUNCtion:PEXCursion

CALCulate<x>:[SELected]:FUNCtion:EXECute

CALCulate<x>:[SELected]:FUNCtion:DATA?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

Select one of the following arguments:

| Argument | Description |
|---|---|
| PTPeak | Difference between max. and min. values |
| STDEV | Standard deviation |
| MEAN | Mean value |
| MAXimum | Max. value |
| MINimum | Min. value |
| PEAK | Max. positive or min. negative peak |
| APEak | All peaks |
| ATARget | All targets |

**Returns** See Arguments.

**Examples** `CALC1:SEL:FUNC:TYPE STDEV` sets the analysis type to standard deviation for the active trace on channel 1.

`CALC2:SEL:FUNC:TYPE?` may return MEAN, which is the analysis type set for the active trace on channel 2.

## CALCulate<x>:[SELected]:LIMit:DATA

Set or query the limit table for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:DATA { numeric 1 | numeric
1+(n*5)-4 | numeric 1+(n*5)-3 | numeric 1+(n*5)-2 | numeric
1+(n*5)-1 | numeric 1+(n*5) }
CALCulate<x>:[SELected]:LIMit:DATA?
```

**Related Commands** CALCulate<x>:[SELected]:LIMit:[STATe]

MMEMory:STORe:LIMit

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

Select one of the following arguments for the limit table:

| Argument | Description |
|---|---|
| numeric1 | No. of limit lines (1 to 100) |
| numeric1+(n*5)-4 | Type of line can be: 0 (off), 1 (upper limit), or 2 (lower limit). |
| numeric1+(n*5)-3 | Supposed range is based on UI: -200 to 200 of start point of line n. |
| numeric1+(n*5)-2 | X coordinate of end point of line n. |
| numeric1+(n*5)-1 | Supposed range is based on UI: -200 to 200 of start point of line n. |
| numeric1+(n*5) | Y coordinate of end point of line n. |

Range: 3e5 to 6e9

**Returns** See Arguments.

**Examples**

`CALC1:SEL:LIM:DATA 1,1,3e5,6e9,-10,-20` creates an upper limit line with a stimulus range from 3e5 to 6e9 and a beginning and ending response from -10 dB to -20 dB for channel 1.

`CALC1:SEL:LIM:DATA?` may return 1, 1, 300000, 6e+9, -10, -20, which means an upper limit line with a stimulus range from 300000 to 6e+9, and a beginning and an ending response from -10 dB to -20 dB.

## CALCulate<x>:[SELected]:LIMit:DISPlay:[STATe]

Set or query the state of the display for the limit line defined in `CALCulate<x>:[SELected]:LIMit:DATA` for the enabled limit test set by `CALCulate<x>:[SELected]:LIMit:[STATe]` for the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:DISPlay:[STATe] <value>
CALCulate<x>:[SELected]:LIMit:DISPlay:[STATe]?
```

**Related Commands**

CALCulate<x>:[SELected]:LIMit:DATA

CALCulate<x>:[SELected]:LIMit:[STATe]

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns**

1 means the limit line is turned on.

0 means the limit line is turned off.

**Examples**

`CALC1:SEL:LIM:DISP:STAT 1` enables the limit line for the active trace on channel 1.

`CALC4:SEL:LIM:DISP:STA?` may return 0, which means the limit line is disabled for the active trace on channel 4.

## CALCulate<x>:[SELected]:LIMit:FAIL? (Query Only)

Query the limit test result for the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:FAIL?
```

**Related Commands**

CALCulate<x>:[SELected]:LIMit:[STATe]

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

0 means the limit test passed.

1 means the limit test failed.

**Examples**

`CALC1:SEL:LIM:FAIL?` may return 0, which means the limit test passed for the active trace on channel 1.

`CALC1:SEL:LIM:FAIL?` may return 1, which means the limit test failed for the active trace on channel 1.

## CALCulate<x>:[SELected]:LIMit:OFFSet:AMPLitude

Set or query the amplitude offset value (in Hz) for the limit line test of the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:OFFSet:AMPLitude <value>
CALCulate<x>:[SELected]:LIMit:OFFSet:AMPLitude?
```

**Related Commands**

CALCulate<x>:[SELected]:LIMit:[STATe]

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the amplitude offset value. Range: -5e8 to 5e8 Hz.

**Returns**

`<NRf>`

**Examples**

`CALC1:SEL:LIM:OFFS:AMPL 3e8` sets the amplitude offset to 0.3 GHz for the active trace on channel 1.

`CALC3:SEL:LIM:OFFS:AMPL?` may return -4e8, which means -0.4 GHz is the amplitude offset value set for the active trace on channel 3.

## CALCulate<x>:[SELected]:LIMit:OFFSet:MARKer (No Query Form)

Set the amplitude offset value to the active marker value for the limit line test of the active trace on the specified channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:OFFSet:MARKer
```

**Related Commands** CALCulate<x>:[SELected]:LIMit:[STATe]

**Arguments** None

**Examples** `CALC1:SEL:LIM:OFFS:MARK` sets the amplitude offset value to the current active marker value on channel 1.

## CALCulate<x>:[SELected]:LIMit:OFFSet:STIMulus

Set or query the stimulus offset value (in Hz) for the limit line test of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:OFFSet:STIMulus <value>
CALCulate<x>:[SELected]:LIMit:OFFSet:STIMulus?
```

**Related Commands** CALCulate<x>:[SELected]:LIMit:[STATe]

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the stimulus offset value. Range: -1e12 to 1e12 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:LIM:OFFS:STIM 1e8` sets 100 MHz for the value of the limit line stimulus offset for the active trace on channel 1.

`CALC3:SEL:LIM:OFFS:STIM?` may return -2e8, which means -200 MHz is the limit line stimulus offset value set for the active trace on channel 3.

## CALCulate<x>:[SELected]:LIMit:REPort:ALL? (Query Only)

Query the limit test result for all measurement points of the active trace on the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:LIMit:REPort:ALL?`

**Related Commands** CALCulate<x>:[SELected]:LIMit:REPort:DATA?

CALCulate<x>:[SELected]:LIMit:REPort:POINts?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

The return is the data array of the number of measurement points *n* in the active trace.

`[n*4-3],[n*4-2],[n*4-1],[n*4]` where:

| Argument | Description |
|---|---|
| n*4–3 | Stimulus value |
| n*4–2 | Test result: 0 (fail), 1 (pass), –1 (no test) |
| n*4–1 | Upper limit value |
| n*4 | Lower limit value |

**Examples** `CALC1:SEL:LIM:REP:ALL?` may return 300000, 0, -100, 0, 5e9, 1, -100, 0, if there are only two measurement points. The first group indicates that the first point at 3e5 failed the -100 dB value by having a higher amplitude response (upper limit). The second point at 5e9 passed because its amplitude was lower than -100 dB.

## CALCulate<x>:[SELected]:LIMit:REPort:DATA? (Query Only)

Query the stimulus values of all measurement points that failed the limit test for the active trace on the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:LIMit:REPort:DATA?`

**Related Commands** CALCulate<x>:[SELected]:LIMit:REPort:ALL?

CALCulate<x>:[SELected]:LIMit:REPort:POINts?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`[numeric1],[numeric2],...[numeric`*n*`]`, where

`numeric 1` is the stimulus value at measurement point 1 that failed the limit test (Hz).

**Examples** `CALC1:SEL:LIM:REP:DATA?` may return 1e9, 2e9, 3e9, 4e9, 5e9. Assuming the stimulus range is in Hz, the five measurement points failed the limit test at the stimulus values 1 GHz, 2 GHz, 3 GHz, 4 GHz, and 5 GHz of the active trace on channel 1.

## CALCulate<x>:[SELected]:LIMit:REPort:POINts? (Query Only)

Query the number of measurement points that failed the limit test for the active trace on the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:LIMit:REPort:POINts?`

**Related Commands** CALCulate<x>:[SELected]:LIMit:REPort:ALL?

CALCulate<x>:[SELected]:LIMit:REPort:DATA?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** `CALC1:SEL:LIM:REP:POIN?` may return 35, which is the number of measurement points that failed the limit test for the active trace on channel 1.

## CALCulate<x>:[SELected]:LIMit:[STATe]

Set or query the status of the limit test for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:LIMit:[STATe] <value>
CALCulate<x>:[SELected]:LIMit:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the limit test is turned on.

0 means the limit test is turned off.

**Examples** `CALC1:SEL:LIM:STAT 1` enables the limit test for the active trace on channel 1.

`CALC2:SEL:LIM:STAT?` may return 0, which means the limit test is disabled for the active trace on channel 2.

## CALCulate<x>:[SELected]:MARKer:Y? (Query Only)

Query the marker response value for the specified marker on the active trace for the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer:Y?
```

**Related Commands** CALCulate<x>:[SELected]:MARKer:Y?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`([numeric1],[numeric2])`, where:

`[numeric1]::=<NRf>` is the primary response value.

`[numeric2]::=<NRf>` is the secondary response value. This value is always 0, unless the plot format is Smith Chart or Polar.

**Examples** `CALC1:SEL:MARK1:Y?` may return -109.907,0 for the marker response value of marker 1 for the active trace on channel 1, if the format for channel 1 is set to Logarithmic Magnitude.

## CALCulate<x>:[SELected]:MARKer<y>:[ACTivate] (No Query Form)

Set the specified marker of the active trace for the specified channel.

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:MARKer<y>:[ACTivate]`

**Related Commands** CALCulate<x>:[SELected]:MARKer<y>:[STATe]

**Arguments** None

**Examples** `CALC1:SEL:MARK2:ACT` sets marker 2 as the active marker of the active trace on channel 1.

## CALCulate<x>:[SELected]:MARKer<y>:BWIDth:DATA? (Query Only)

Query the result value(s) for the bandwidth search for the specified marker of the active trace for the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:MARKer<y>:BWIDth:DATA?`

**Related Commands** CALCulate<x>:[SELected]:MARKer<y>:BWIDth:[STATe]

**Returns** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

***NOTE.*** *The query in this command may return nothing if it does not find any data in bandwidth (Hz).*

`[numeric1],[numeric2],[numeric3],[numeric4],`

where:

`[numeric1]::=<NRf>` is the Bandwidth in Hz.

`[numeric2]::=<NRf>` is the Center Frequency in Hz.

`[numeric3]::=<NRf>` is the Q value.

`[numeric4]::=<NRf>` is the Loss in dB.

**Examples**

`CALC1:SEL:MARK2:BWID:DATA?` may return 1.27521e+09, 3.11222e+09, 2.44056, 3, which means: the bandwidth is 1.27521 GHz, the center frequency is 3.1 GHz, the Q value is 2.44056, and the loss is 3 dB, which are the bandwidth search results for marker 2 of the active trace for channel 1.

## CALCulate<x>:[SELected]:MARKer<y>:BWIDth:[STATe]

Set or query the state of the display of the bandwidth search result for the specified marker of the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:BWIDth:[STATe] <value>
CALCulate<x>:[SELected]:MARKer<y>:BWIDth:[STATe]?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:BWIDth:THReshold

CALCulate<x>:[SELected]:MARKer<y>:BWIDth:DATA?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns**

1 means the bandwidth search results are displayed.

0 means the bandwidth search results are hidden.

**Examples**

`CALC1:SEL:MARK1:BWID:STAT 1` enables the bandwidth search results for marker 1 of the active trace for channel 1.

`CALC5:SEL:MARK4:BWID:STAT?` may return 0, which means the bandwidth search results are disabled for marker 4 of the active trace for channel 5.

## CALCulate<x>:[SELected]:MARKer<y>:BWIDth:THReshold

Set or query the bandwidth definition value for the specified marker of the active trace for the specified channel

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:BWIDth:THReshold <value>
CALCulate<x>:[SELected]:MARKer<y>:BWIDth:THReshold?
```

**Related Commands** CALCulate<x>:[SELected]:MARKer<y>:BWIDth:[STATe]

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the bandwidth definition value. Range: -5e8 to 5e8 (Hz).

**Returns** `<NRf>`

**Examples** `CALC1:SEL:MARK2:BWID:THR 4e8` sets the bandwidth definition value to 0.4 GHz for marker 2 of the active trace on channel 1.

`CALC1:SEL:MARK2:BWID:THR?` may return -3e8, which means -0.3 GHz is the bandwidth definition value set for marker 4 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:COUPle

Set or query state of the marker coupling for the specified marker between traces on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:COUPle <value>
CALCulate<x>:[SELected]:MARKer<y>:COUPle?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns** 1 means marker coupling is turned on.

0 means marker coupling is turned off.

**Examples**

`CALC1:SEL:MARK1:COUP 1` enables marker coupling between traces of marker 1 for channel 1.

`CALC2:SEL:MARK4:COUP?` may return 0, which means marker coupling is disabled between traces of marker 4 for channel 2.

## CALCulate<x>:[SELected]:MARKer<y>:DISCrete

Set or query the state of the marker discrete mode for the selected marker of the active trace for the specified channel. In the discrete mode, the marker moves only at measurement points. While in continuous mode, the marker will interpolate data between measurement points.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:DISCrete <value>
CALCulate<x>:[SELected]:MARKer<y>:DISCrete?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

```
<value>::=<boolean>
```

**Returns**

1 means the marker discrete mode is turned on.

0 means the marker discrete mode is turned off.

**Examples**

`CALC1:SEL:MARK1:DISC 1` enables the marker discrete mode for marker 1 of the active trace for channel 1.

`CALC6:SEL:MARK5:DISC?` may return 0, which means the marker discrete mode is disabled for marker 5 of the active trace for channel 6.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle

Set or query the state of the marker search range for the specified marker to coupled or decoupled to all traces or only to the active trace for the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle
<value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns**

1 means the marker search range is coupled to all traces.

0 means the marker search range is decoupled to all traces.

**Examples**

`CALC1:SEL:MARK1:FUNC:DOM:COUP 1` couples the marker search range to all traces for marker 1 on channel 1.

`CALC3:SEL:MARK4:FUNC:DOM:COUP?` may return 0, which means the marker search range is set to decoupled for all traces for marker 4 on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt

Set or query the starting value for the arbitrary marker search range for the specified marker of the active trace for the specified channel. The arbitrary marker search range is set by `CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]`.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt
<value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the starting value for the marker search range (Hz).

**Returns** `<NRf>`

**Examples** `CALC1:SEL:MARK1:FUNC:DOM:STAR 2e8` sets the starting value to 0.2 GHz for the arbitrary marker search range for marker 1 of the active trace for channel 1.

`CALC3:SEL:MARK2:FUNC:DOM:STAR?` may return 4e8, which means 0.4 GH is the starting value of the arbitrary search range set for marker 2 of the active trace for channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]

Set or query an arbitrary or entire sweep range for a marker search on the specified marker of the active trace of on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]
<value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]?
```

**Related Commands** CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns** 1 means the arbitrary range is set for marker search.

0 means the entire sweep range is set for marker search.

**Examples** `CALC1:SEL:MARK1:FUNC:DOM:STAT 1` sets an arbitrary range for a search on marker 1 of the active trace for channel 1.

`CALC3:SEL:MARK4:FUNC:DOM:STAT?` may return 0, which means an entire sweep range is set for a search on marker 4 of the active trace for channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP

Set or query the stopping value for the arbitrary marker search range for the specified marker of the active trace for the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP
<value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STOP?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:COUPle

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:[STATe]

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:DOMain:STARt

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the stopping value for the marker search range (Hz).

**Returns** `<NRf>`

**Examples**

`CALC1:SEL:MARK1:FUNC:DOM:STOP 3e8` sets the stopping value to 0.3 GHz for the marker search range for marker 1 of the active trace for channel 1.

`CALC3:SEL:MARK2:FUNC:DOM:STOP?` may return 2e8, which means 0.2 GHz is the stopping value of the marker search range set for marker 2 of the active trace for channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PEXCursion

Set or query the lower limit of the peak excursion value when executing a peak excursion search for the specified marker of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PEXCursion
<value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PEXCursion?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE

CALCulate<x>:[SELected]:FUNCtion:PPOLarity

CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the lower limit of the peak excursion value. Range: 0 to 5e8 (Hz).

**Returns**

`<NRf>`

**Examples**

`CALC1:SEL:MARK1:FUNC:PEXC 1e8` sets the lower limit to 0.1 GHz for the peak excursion search value for marker 1 of the active trace for channel 1.

`CALC3:SEL:MARK4:FUNC:PEXC?` may return 4e8, which means 0.4 GHz is the lower limit of the peak excursion search value set for marker 4 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PPOLarity

Set or query the polarity for a peak excursion search for the specified marker of the active trace for the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PPOLarity {
POSitive | NEGative | BOTH }
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PPOLarity?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PEXCursion

CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`POSitive` is positive polarity.

`NEGative` is negative polarity.

`BOTH` is positive and negative polarity.

**Returns**

`POSitive` means positive polarity.

`NEGative` means negative polarity.

`BOTH` means positive and negative polarity.

**Examples**

`CALC1:SEL:MARK1:FUNC:PPOL POS` sets positive polarity for the peak excursion search for marker 1 of the active trace on channel 1.

`CALC3:SEL:MARK2:FUNC:PPOL?` may return NEGative, which is the polarity type set for the peak excursion search for marker 2 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TARGet

Set or query a target value for a target search on the specified marker of the active trace for the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TARGet <value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TARGet?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TTRansition

CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute

CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the target value specified for the target search. Range: -5e8 to 5e8 (Hz).

**Returns**

`<NRf>`

**Examples**

`CALC1:SEL:MARK1:FUNC:TARG 2e8` sets the target search value to 0.2 GHz for marker 1 of the active trace for channel 1.

`CALC4:SEL:MARK3:FUNC:TARG?` may return -3e8, which means -0.3 GHz is the target search value set for marker 3 of the active trace for channel 4.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TRACking

Set or query the state of the marker tracking for a marker search for the specified marker of the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TRACking <value>
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TRACking?
```

**Related Commands**

CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute

CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns**

1 means marker tracking is enabled.

0 means marker tracking is disabled.

**Examples**

`CALC1:SEL:MARK1:FUNC:TRAC 1` turns on marker tracking for marker 1 of the active trace on channel 1.

`CALC3:SEL:MARK2:FUNC:TRAC?` may return 0, which means marker tracking is turned off for marker 2 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TTRansition

Set or query the transition type used during a target search for the specified marker on the active trace for the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TTRansition {
POSitive | NEGative | BOTH }
CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TTRansition?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TARGet

CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute

CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`POSitive` is a positive transition.

`NEGative` is a negative transition.

`BOTH` is a positive and negative transition.

**Returns**

`POSitive` means a positive transition is set for a target search.

`NEGative` means a negative transition is set for a target search.

`BOTH` means a positive and negative transition is set for a target search.

**Examples**

`CALC1:SEL:MARK1:FUNC:TTR POS` sets the target search to look for positive transitions for marker 1 of the active trace on channel 1.

`CALC4:SEL:MARK2:FUNC:TTR?` may return NEGative, which means a negative transition is set for a target search on marker 2 of the active trace on channel 4.

## CALCulate<x>:[SELected]:MARKer<y>:NOTCh:DATA? (Query Only)

Query the results from a notch search using the specified marker of the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:NOTCh:DATA?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:NOTCh:[STATe]

CALCulate<x>:[SELected]:MARKer<y>:NOTCh:THReshold

**Returns**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`[numeric1],[numeric2],[numeric3],[numeric4]`, where:

`[numeric1]::=<NRf>` is the Bandwidth in Hz.

`[numeric2]::=<NRf>` is the Center Frequency in Hz.

`[numeric3]::=<NRf>` is the Q value.

`[numeric4]::=<NRf>` is the Loss in dB

**Examples**

`CALC1:SEL:MARK1:NOTC:DATA?` may return the results for the notch search on marker 1 of the active trace for channel 1.

## CALCulate<x>:[SELected]:MARKer<y>:NOTCh:[STATe]

Set or query the notch search for the specified marker of the active trace for the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:NOTCh:[STATe] <value>
CALCulate<x>:[SELected]:MARKer<y>:NOTCh:[STATe]?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:NOTCh:DATA?

CALCulate<x>:[SELected]:MARKer<y>:NOTCh:THReshold

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns**

1 means the notch search is enabled.

0 means the notch search is disabled.

**Examples**

`CALC1:SEL:MARK1:NOTC:STAT 1` turns on the notch search for marker 1 of the active trace on channel 1.

`CALC2:SEL:MARK2:NOTC:STAT?` may return 0, which means the notch search is turned off for marker 2 of the active trace on channel 2.

## CALCulate<x>:[SELected]:MARKer<y>:NOTCh:THReshold

Set or query the notch value definition during a notch search for the specified marker of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:NOTCh:THReshold <value>
CALCulate<x>:[SELected]:MARKer<y>:NOTCh:THReshold?
```

**Related Commands** CALCulate<x>:[SELected]:MARKer<y>:NOTCh:DATA?

CALCulate<x>:[SELected]:MARKer<y>:NOTCh:[STATe]

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the notch value definition. Range: -5e8 to 5e8 Hz.

**Returns** `<NRf>`

**Examples** `CALC1:SEL:MARK1:NOTC:THR 1e8` sets the notch value definition to 0.1 GHz for the notch search on marker 1 of the active trace on channel 1.

`CALC3:SEL:MARK2:NOTC:THR?` may return -2e8, which means -0.2 GHz is the notch value definition set for the notch search for marker 2 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:REFerence:[STATe]

Set or query the state of reference marker mode for the specified marker on the active trace for the specified channel. If the reference marker is not enabled, this command will then enable the reference marker.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:REFerence:[STATe] <value>
CALCulate<x>:[SELected]:MARKer<y>:REFerence:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 9).

```
<value>::=<boolean>
```

**Returns** 1 means the reference marker mode is turned on.

0 means the reference marker mode is turned off.

**Examples** `CALC1:SEL:MARK1:REF:STAT 1` enables the reference marker mode for marker 1 of the active trace on channel 1.

`CALC2:SEL:MARK2:REF:STAT?` may return 0, which means the reference marker mode is disabled for marker 2 on the active trace for channel 2.

## CALCulate<x>:[SELected]:MARKer<y>:SET (No Query Form)

Set the value of the specified marker position to the specified value of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:SET { STARt | STOP |
CENTer | RLEVel | DELay }
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

Choose a marker position from the list below:

| Argument | Description |
|---|---|
| STARt | Set sweep start value to stimulus value at marker position. |
| STOP | Set sweep stop value to stimulus value at marker position. |
| CENTer | Set sweep center value to stimulus value at marker position. |
| RLEVel | Set reference line value to the response value at marker position. |
| DELay | Set electrical delay value to response value at marker position. |

**Examples** `CALC1:SEL:MARK1:SET CENT` takes the stimulus value of marker 1 located on the active trace and sets this value to the sweep center value for channel 1. If marker 1 on the active trace has a stimulus value of 4 GHz, the center frequency for channel 1 becomes 4 GHz.

## CALCulate<x>:[SELected]:MARKer<y>:[STATe]

Set or query the state of the display of the specified marker of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:[STATe] <value>
CALCulate<x>:[SELected]:MARKer<y>:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<boolean>`

**Returns** 1 means the specified marker of the active trace is turned on.

0 means the specified marker of the active trace is turned off.

**Examples** `CALC1:SEL:MARK1:STAT 1` enables marker 1 of the active trace for channel 1.

`CALC3:SEL:MARK2:STAT?` may return 0, which means marker 2 is disabled for the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>:X

Set or query the marker stimulus value for the specified marker on the active trace for the specified channel. If the specified frequency is outside the range, the minimum or maximum frequency is set accordingly.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>:X <value>
CALCulate<x>:[SELected]:MARKer<y>:X?
```

**Related Commands** CALCulate<x>:[SELected]:MARKer:Y?

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<value>::=<NRf>` is the marker stimulus value. Range: Sweep start to sweet stopping values (Hz).

**Returns** `<NRf>`

**Examples** `CALC1:SEL:MARK1:X 2e8` sets 0.2 GHz for the marker stimulus value of marker 1 of the active trace on channel 1.

`CALC3:SEL:MARK2:X?` may return 3e8, which means 0.3 GHz is the marker stimulus value set for marker 2 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE

Set or query the type of search used in `CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:EXECute` for the specified marker of the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE { MAXimum |
MINimum | PEAK | LPEak | RPEak | TARGet | LTARget | RTARget
}
CALCulate<x>:[SELected]:MARKer<y>FUNCtion:TYPE?
```

**Related Commands**

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TTRansition

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TARGet

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PPOLarity

CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:PEXCursion

CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

`<string>` where the analysis type is:

| Argument | Description |
|---|---|
| Maximum | Find max. peak |
| MINimum | Find min. peak. |
| PEAK | Find peak. |
| LPEak | Find peaks to left of marker position. |
| RPEak | Find peaks to right of marker position. |
| TARGet | Find target. |

| Argument | Description |
|---|---|
| LTARget | Find targets to left of marker position. |
| RTARget | Find targets to right of marker position. |

**Returns** See Arguments.

**Examples** `CALC1:SEL:MARK1:FUNC:TYPE MAX` sets the search type to maximum peak for marker 1 of the active trace on channel 1.

`CALC3:SEL:MARK2:FUNC:TYPE?` may return MIN, which means minimum peak is the search type set for marker 2 of the active trace on channel 3.

## CALCulate<x>:[SELected]:MATH:FUNCtion

Set or query the data trace method (math operation) for the data trace based on the stored memory trace `CALCulate<x>:[SELected]:MATH:MEMorize` for the active trace on the specified channel. The data trace shows the math result.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MATH:FUNCtion { NORMal | DIVide |
MULTiple | SUBTract | ADD }
CALCulate<x>:[SELected]:MATH:FUNCtion?
```

**Related Commands** CALCulate<x>:[SELected]:MATH:MEMorize

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

Select one of these data trace operations:

| Argument | Description |
|---|---|
| NORMal | Data Only, No Math |
| DIVide | Data/Memory |
| MULTiply | Data x Memory |
| SUBTract | Data - Memory |
| ADD | Data + Memory |

`Data` is the measurement data.

`Memory` is the data stored in the memory trace.

**Returns** See Arguments.

**Examples** `CALC1:SEL:MATH:FUNC MULT` sets the `Data x Memory` operation type for the active trace on channel 1.

`CALC3:SEL:MATH:FUNC?` may return DIV, which means a `Data/Memory` is the operation type set for the active trace on channel 3.

## CALCulate<x>:[SELected]:MATH:MEMorize (No Query Form)

Copy the measurement data to the memory trace for the active trace on the specified channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:MATH:MEMorize
```

**Examples** `CALC1:SEL:MARK1:MATH:MEM` copies the measurement data to the memory trace for the active trace on channel 1.

## CALCulate<x>:[SELected]:PHASe

Set or query the phase unit used for data formats `PHASe, UPHase, and PPHase (CALCulate<x>:[SELected]:FORMat)` for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:PHASe { DEGrees | RADians }
CALCulate<x>:[SELected]:PHASe?
```

**Related Commands** CALCulate<x>:[SELected]:FORMat

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<string>` where

`DEGrees` sets the phase unit to degrees.

`RADians` sets the phase unit to radians.

**Returns** `DEGrees` means the phase unit is set to degrees.

`RADians` means the phase unit is set to radians.

**Examples** `CALC1:SEL:PHAS RAD` sets radians for the phase unit for the active trace on channel 1.

`CALC2:SEL:PHAS?` may return DEGrees, which means degrees is the phase unit set for the active trace on channel 2.

## CALCulate<x>:[SELected]:RLIMit:DATA

Set or query the value for the ripple limit table for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:RLIMit:DATA { numeric 1 | numeric
[1+n*4-3] | numeric [1+n*4-2] | numeric [1+n*4-1] | numeric
[1+n*4] }
CALCulate<x>:[SELected]:RLIMit:DATA?
```

**Related Commands** MMEMory:STORe:RLIMit

**Returns** **Command variables.** <x> is the channel number (1 to 16).

This query returns an array of data for `1+n*4`, where *n* is the number of limit lines (1 to 12), where:

`[1]` indicates the number of ripple limit bands (0 to *n* limit lines).

`[1+n*4-3]` indicates the type of ripple limit band *n* (0= OFF; 1= ON).

`[1+n*4-2]` indicates the starting stimulus value of band *n*.

`[1+n*4-1]` indicates the ending stimulus value of band *n*.

`[1+n*4]` indicates the maximum ripple (dB) range of band *n*.

**Examples** `CALC1:SEL:RLIM:DATA 2,1,1e5,6e9,10,0,2e,5e7,7` sets the first ripple line as an active ripple limit line for the active trace on channel 1, with a range from 1e5 to 6e9 at 10 dB. The second ripple line is inactive with a range from 2e7 to 5e7 at 7 dB.

`CALC:SEL:RLIM:DATA 0` clears the ripple limit table.

`CALC1:SEL:RLIM:DATA?` may return 2, 1, 100000, 6e+9, 10, 0, 2e+07, 5e+07, 7, which is the first ripple line set as an active ripple limit line for the active trace on channel 1.

## CALCulate<x>:[SELected]:RLIMit:DISPlay:LINE

Set or query the state of the display of the ripple limit line for the active trace on the specified channel, by setting both the ripple limit test `CALCulate<x>:[SELected]:RLIMit:[STATe]` and ripple limit table `CALCulate<x>:[SELected]:RLIMit:DATA`.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:RLIMit:DISPlay:LINE <value>
CALCulate<x>:[SELected]:RLIMit:DISPlay:LINE?
```

**Related Commands** CALCulate<x>:[SELected]:RLIMit:DATA

CALCulate<x>:[SELected]:RLIMit:[STATe]

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the ripple limit line is turned on.

0 means the ripple limit line is turned off.

**Examples** `CALC1:SEL:RLIM:DISP:LINE 1` enables the display of the ripple limit line for the active trace on channel 1.

`CALC2:SEL:RLIM:DISP:LINE?` may return 0, which means the display of the ripple limit line is disabled for the active trace on channel 2.

## CALCulate<x>:[SELected]:RLIMit:DISPlay:[SELect]

Set or query the ripple limit band that shows the value of the ripple test result for the active trace on the specified channel. Set the display value for the ripple limit band with `CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue`.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:RLIMit:DISPlay:[SELect] <value>
CALCulate<x>:[SELected]:RLIMit:DISPlay:[SELect]?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the ripple limit band. Range: 1 to 12.

**Returns**

`<NR1>`

**Examples**

`CALC1:SEL:RLIM:DISP:SEL 3` sets the ripple limit band to 3 from the value of the ripple test result for the active trace on channel 1. This example requires at least 3 ripple limit lines set on the ripple limit table.

`CALC3:SEL:RLIM:DISP:SEL?` may return 2, which is the ripple limit band from the value set for the ripple test result for the active trace on channel 3.

## CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue

Set or query the display type for the ripple value of the active trace on the specified channel. Change between ripple lines to display the ripple value with `CALCulate<x>:[SELected]:RLIMit:DISPlay:[SELect]`.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue { OFF |
ABSolute | MARgin }
CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue?
```

**Related Commands**

CALCulate<x>:[SELected]:RLIMit:DISPlay:VALue

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`OFF` hides the ripple value for the active trace.

`ABSolute` is the display type of the ripple value.

`MARgin` sets a margin for the display of the ripple value.

**Returns**

`OFF` means the ripple value is hidden on the active trace.

`ABSolute` means the ripple value display is absolute.

`MARgin` means a margin is set for the display of the ripple value.

**Examples** `CALC3:SEL:RLIM:DISP:VAL ABS` sets absolute as the display type of the ripple value of the active trace for channel 3.

`CALC2:SEL:RLIM:DISP:VAL?` may return OFF, which means the display of the ripple value of the active trace is disabled for channel 2.

## CALCulate<x>:[SELected]:RLIMit:FAIL? (Query Only)

Query all ripple value results for the ripple limit test of the active trace for the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:RLIMit:FAIL?`

**Returns** **Command variables.** <x> is the channel number (1 to 16).

1 means the ripple test failed.

0 means the ripple test passed.

**Examples** `CALC1:SEL:RLIM:FAIL?` may return 1, which means the ripple test failed for the active trace on channel 1.

`CALC1:SEL:RLIM:FAIL?` may return 0, which means the ripple test passed for the active trace on channel 1.

## CALCulate<x>:[SELected]:RLIMit:REPort:DATA? (Query Only)

Query all ripple value results for the ripple limit test of the active trace for the specified channel.

**Group** Calculate commands

**Syntax** `CALCulate<x>:[SELected]:RLIMit:REPort:DATA?`

**Returns** **Command variables.** <x> is the channel number (1 to 16).

This query returns an array of data of 1+n*3, where *n* is the number of limit lines (1 to 12):

`[1]` is the number of ripple limit lines.

`[1+n*3-2]` is the index of the ripple limit band.

`[1+n*3-1]` is the ripple value.

`[1+n*3]` is the result of the ripple test (0= pass; 1= fail).

**Examples** `CALC1:SEL:RLIM:REP:DATA?` may return 2, 0, 36.0053, 0, 1, 12.619, 1, where 2 ripple bands were found. The first band at index 0 has a ripple value of 36.0053, and passes while the second band at index 1 with a ripple value of 12.619 fails for the active trace on channel 1.

## CALCulate<x>:[SELected]:RLIMit:[STATe]

Set or query the state of the display of the ripple test for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:RLIMit:[STATe] <value>
CALCulate<x>:[SELected]:RLIMit:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 means the ripple test is disabled.

1 means the ripple test is enabled.

**Examples** `CALC1:SEL:RLIM:STAT 1` enables the ripple test for the active trace on channel 1.

`CALC2:SEL:RLIM:STAT?` may return 0, which means the ripple test is disabled for the active trace on channel 2.

## CALCulate<x>:[SELected]:SMOOthing:APERture

Set or query the smoothing aperture percentage of the sweep span for the active trace on the specified channel.

**Group** Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:SMOOthing:APERture <value>
CALCulate<x>:[SELected]:SMOOthing:APERture?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the smoothing aperture percentage of the sweep span. Range: 0.05 to 25%.

**Returns**

`<NRf>`

**Examples**

`CALC1:SEL:SMO:APER 10` sets 10% for the percentage of the smoothing aperture of the sweep span for the active trace on channel 1

`CALC2:SEL:SMO:APER?` may return 12, which means 12% is the percentage of the smoothing aperture of the sweep span set for the active trace on channel 2.

## CALCulate<x>:[SELected]:SMOOthing:[STATe]

Set or query the smoothing aperture for the active trace on the specified channel.

**Group**

Calculate commands

**Syntax**

```
CALCulate<x>:[SELected]:SMOOthing:[STATe] <value>
CALCulate<x>:[SELected]:SMOOthing:[STATe]?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns**

0 means the smoothing aperture is disabled.

1 means the smoothing aperture is enabled.

**Examples**

`CALC1:SEL:SMO:STAT 1` enables the smoothing aperture for the active trace on channel 1.

`CALC2:SEL:SMO:STAT?` may return 0, which means the smoothing aperture is disabled for the active trace on channel 2.

## CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute (No Query Form)

Execute the marker search set by `CALCulate<x>:[SELected]:MARKer<y>:FUNCtion:TYPE` for the specified marker of the active trace on the specified channel.

**Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 10).

**Group** Calculate commands

**Syntax** `CALCulate<x>[SELected]:MARKer<y>:FUNCtion:EXECute`

**Related Commands** CALCulate<x>:[SELected]:FUNCtion:TYPE

**Arguments** None

**Examples** `CALC1:SEL:MARK1:FUNC:EXEC` executes the marker search for marker 1 of the active trace for channel 1. If the marker search type is set to MAX, marker 1 then moves to the maximum peak within the given search range.

## *CLS (No Query Form)

Clear these analyzer status data structures.

- Event Queue
- Status Byte Register (except the MAV bit)
- Standard Event Status Register (SESR)

If the *CLS command immediately follows an <EOI>, the Output Queue and MAV bit (Status Byte Register bit 4) are also cleared. MAV indicates the information is in the output queue. The device clear (DCL) GPIB control message clears the output queue and thus MAV.

*CLS does not clear the output queue or MAV.

*CLS can suppress a Service Request that is to be generated by an *OPC. This happens if a hardcopy output or single sequence acquisition operation is still being processed when the *CLS command is executed.

For a complete discussion of these registers and bits, and of event handling in general, refer to the Status and Events section.

**Group** IEEE common commands

**Syntax** `*CLS`

**Related Commands** *ESE, *ESR?, *SRE, *STB?

**Examples** `*CLS` clears the status data structures in the analyzer.

## DISPlay:ANNotation:FREQuency:[STATe]

Set or query the state of the display of the frequency for all windows.

**Group** Display commands

**Syntax**

```
DISPlay:ANNotation:FREQuency:[STATe] <value>
DISPlay:ANNotation:FREQuency:[STATe]?
```

**Arguments** `<value>::=<boolean>`

**Returns** 0 means the frequency is hidden.

1 means the frequency shows.

**Examples** `DISP:ANN:FREQ:STAT 0` turns off the display of the frequency for all windows.

`DISP:ANN:FREQ:STAT?` may return 1, which means the display of the frequency is turned on for all windows.

## DISPlay:CHANnel<x>:[ACTivate] (No Query Form)

Set the channel for the active graph set by `DISPlay:GRAPh<x>:[ACTivate]`.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Display commands

**Syntax** `DISPlay:CHANnel<x>:[ACTivate]`

**Related Commands** DISPlay:GRAPh<x>:[ACTivate]

**Arguments** None

**Examples** `DISP:CHAN2:ACT` sets channel 2 as the active channel for the active graph.

`DISP:CHAN1:ACT` sets channel 1 as the active channel for the active graph.

## DISPlay:CHANnel<x>:ANNotation:MARKer:ALIGn:[STATe]

Set or query the status of the alignment for all readouts for markers on trace 1 for the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:ANNotation:MARKer:ALIGn:[STATe] <value>
DISPlay:CHANnel<x>:ANNotation:MARKer:ALIGn:[STATe]?
```

**Related Commands** DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:X

DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:Y

DISPlay:CHANnel<x>:ANNotation:MARKer<y>:SINGle:[STATe]

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

```
<value>::=<boolean>
```

**Returns** 0 means you can place the marker information for each trace in a different part of the graph

1 means all marker information aligns to the location under the marker information from trace 1.

**Examples** `DISP:CHAN2:ANN:MARK:ALIG:STAT 1` aligns all marker information to the location under the marker information from trace 1 for channel 2.

`DISP:CHAN3:ANN:MARK:ALIG:STAT?` may return 0, which means you can position the marker information for each trace in a different part of the graph for channel 3.

## DISPlay:CHANnel<x>:ANNotation:MARKer<y>:SINGle:[STATe]

Set or query the display of marker information for all traces or only on the active trace for the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:ANNotation:MARKer<y>:SINGle:[STATe]
<value>
DISPlay:CHANnel<x>:ANNotation:MARKer<y>:SINGle:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the marker number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 means the marker information shows for all traces.

1 means the marker information only shows for the active trace.

**Examples** `DISP:CHAN2:ANN:MARK:SING:STAT 1` enables the display of the marker information to only show for trace 1 on channel 2.

`DISP:CHAN3:ANN:MARK:SING:STAT?` may return 0, which means the display of the marker information is enabled for all traces on channel 3.

## DISPlay:CHANnel<x>:LABel

Set or query the state of the display of the vertical axis label on the specified graph and channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:LABel <value>
DISPlay:CHANnel<x>:LABel?
```

**Related Commands** DISPlay:GRAPh<x>:[ACTivate]:LABel

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 means the vertical axis label is hidden.

1 means the vertical axis label shows.

**Examples** `DISP:CHAN2:LAB 1` enables the display of the vertical axis label for channel 2.

`DISP:CHAN3:LAB?` may return 0, which means the display of the vertical axis label is disabled for channel 3.

## DISPlay:CHANnel<x>:MAXimize

Maximize the active trace of the specified channel. Query the maximization status of the active trace for the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:MAXimize <value>
DISPlay:CHANnel<x>:MAXimize?
```

**Arguments**

```
<value>::=<boolean>
```

**Returns** 0 means that the maximization of the active trace is turned off.

1 means that the maximization of the active trace is turned on.

**Examples** `DISP:CHAN2:MAX ON` maximizes the active trace of channel 2.

`DISP:CHAN1:MAX?` may return 0, which means the maximization of the active trace of channel 1 is disabled.

## DISPlay:CHANnel<x>:SPLit

Set or query the display layout for traces on the selected channel.

**Group** Display commands

**Syntax**

***NOTE.*** *These arguments D123__ABC , D1234__9ABC , D1234__DEFG require 2 underscores, unlike other arguments.*

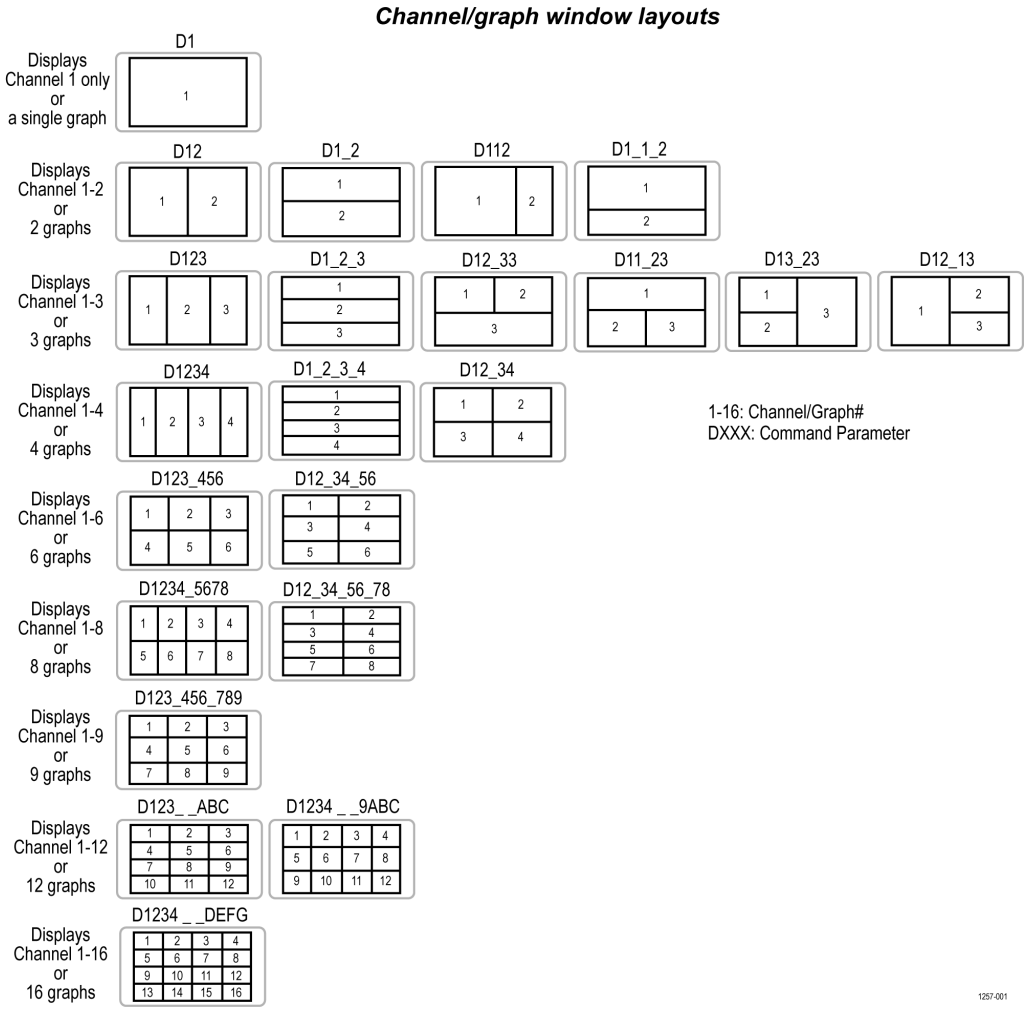
```
DISPlay:CHANnel<x>:SPLit { D1 | D12 | D1_2 | D112 | D1_1_2
| D123 | D1_2_3 | D12_33 | D11_23 | D13_23 | D12_13 | D1234
```

```
| D1_2_3_4 | D12_34 | D123_456 | D12_34_56 | D1234_5678 |
D12_34_56_78 | D123_456_789 | D123__ABC | D1234__9ABC |
D1234__DEFG }
DISPlay:CHANnel<x>:SPLit?
```

**Related Commands** DISPlay:SPLit

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

Select one of the arguments. Each argument correlates to one of these graph display layouts:




**Returns** See Arguments.

**Examples**

`DISP:CHAN2:SPL D12_34` sets the display of the trace layout to show channel 1-3 or 3 graphs on channel 2.

`DISP:CHAN3:SPL?` may return D12_34, which means the trace layout is set for channel 1-3 or 3 graphs on channel 3.

## DISPlay:CHANnel<x>:TITLe:DATA

Set or query the label title for the specified channel.

**Group**

Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TITLe:DATA <string>
DISPlay:CHANnel<x>:TITLe:DATA?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<string>::=<NRf>` is the label title.

**Returns**

`<NRf>`

**Examples**

`DISP:CHAN2:TITL:DATA "BPF"` sets BPF as the label title for channel 2.

`DISP:CHAN3:TITL:DATA?` may return TEST, which is the label title set for channel 3.

## DISPlay:CHANnel<x>:TITLe:[STATe]

Set or query the state of the display of the label title for the specified channel.

**Group**

Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TITLe:[STATe] <value>
DISPlay:CHANnel<x>:TITLe:[STATe]?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 hides the label title.

1 shows the label title.

**Examples** `DISP:CHAN1:TITL:STAT 1` enables the display of the label title for channel 1.

`DISP:CHAN2:TITL:STAT?` may return 0, which means the display of the label title is disabled for channel 2.

## DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:X

Set or query the display position of the marker value on the X-axis, using a percentage of the width of the display span for the specified trace on the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:X
<value>
DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:X?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16); <z> is the marker percentage number (-15 to 100%).

`<value>::=<NR1>` is the display position of the marker value on the X-axis expressed as a percentage. Range: -15 to 100%.

**Returns** `<NR1>`

**Examples** `DISP:WIND1:TRAC3:ANN:MARK:POS:X 20` sets the position to 20% for the marker value on the X-axis for trace 3 on channel 1.

`DISP:WIND2:TRAC4:ANN:MARK:POS:X?` may return -5, which means -5% is the position of the marker value on the X-axis for trace 4 on channel 2.

## DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:Y

Set or query the position of the marker value on the Y-axis, using a percentage of the width of the display span for the specified trace on the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:Y
<value>
DISPlay:CHANnel<x>:TRACe<y>:ANNotation:MARKer<z>:POSition:Y?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16); <z> is the marker percentage number.

`<value>::=<NR1>` is the display position of the marker value on the Y-axis, which is expressed as a percentage. Range: -15 to 100%.

**Returns**

`<NR1>`

**Examples**

`DISP:WIND1:TRAC2:ANN:MARK:POS:Y 10` sets the position to 10% for the marker value on the Y-axis for trace 2 on channel 1.

`DISP:WIND2:TRAC3:ANN:MARK:POS:Y?` may return -8, which means -8% is the position of the marker value on the Y-axis for trace 3 on channel 2.

## DISPlay:CHANnel<x>:TRACe<y>:MEMory:[STATe]

Set or query the status for the display of the memory trace for the specified trace on the specified channel. To store the memory trace of the desired trace, use `CALCulate<x>:[SELected]:MATH:MEMorize`.

**Group**

Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:MEMory:[STATe] <value>
DISPlay:CHANnel<x>:TRACe<y>:MEMory:[STATe]?
```

**Related Commands**

CALCulate<x>:[SELected]:MATH:MEMorize

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<boolean>`

**Returns**

0 means the memory trace is turned off.

1 means the memory trace is turned on.

**Examples**

`DISP:WIND2:TRAC1:MEM:STAT 1` enables the display of the memory trace for trace 1 on channel 2.

`DISP:WIND3:TRAC2:MEM:STAT?` may return 0, which means the display of the memory trace is disabled for trace 2 on channel 3.

## DISPlay:CHANnel<x>:TRACe<y>:[STATe]

Set or query the status of the display for the data trace on the specified trace for the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:[STATe] <value>
DISPlay:CHANnel<x>:TRACe<y>:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the data trace is turned on.

0 means the data trace is turned off.

**Examples** `DISP:WIND2:TRAC3:STAT 0` disables the display of data trace 3 on channel 2.

`DISP:WIND2:TRAC4:STAT?` may return 1, which means the display of data trace 4 is enabled for channel 2.

## DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:AUTO

Execute the auto scale function for the specified trace on the specified channel. This function adjusts the value of the reference value and the scale per division to display the trace.

**Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:AUTO
```

**Examples** `DISP:WIND1:TRAC2:Y:SCAL:AUTO` executes the auto scale function for trace 2 on channel 1.

## DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:PDIVision

Set or query the scale per division of the specified trace on the specified channel, when the data format is not Smith or Polar format. Set the full scale value (value of the outermost circumference), if the data format is Smith Chart or Polar formats. If the specified value is outside the allowable setup range, a minimum or maximum value is set.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:PDIVision <value>
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:PDIVision?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<NRf>` is the scale per division value for the specified trace. Range: 1e-18 to 1e8.

**Returns** `<NRf>`

**Examples** `DISP:WIND1:TRAC2:Y:SCALe:PDIV 1e-8` sets 0.1 GHz as the scale per division value of trace 2 on channel 1.

`DISP:WIND2:TRAC3:Y:SCALe:PDIV?` may return 1e10, which means 10 GHz is the scale per division value set for trace 3 on channel 2.

## DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RLEVEL

Set or query the value of the reference division line on the specified trace for the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RLEVEL <value>
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RLEVEL?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<NRf>` is the value of the reference division line. Range: -5e8 to 5e8.

**Returns** `<NRf>`

**Examples** `DISP:WIND1:TRAC2:MEM:Y:SCAL:RLEVEL 1e8` sets to 100 MHz as the value of the reference division line for trace 2 on channel 1.

`DISP:WIND2:TRAC3:MEM:Y:SCAL:RLEVEL?` may return -4e8, which means -400 MHz is the value of the reference division line set for trace 3 on channel 2.

## DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RPOSition

Set or query the position of the reference division line, using its position number (assigned integer starting from the lowest division) on the specified trace for the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RPOSition <value>
DISPlay:CHANnel<x>:TRACe<y>:Y:[SCALe]:RPOSition?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the trace number (1 to 16).

`<value>::=<NR1>` is the position of the reference division line. Range: 0 to number of divisions.

**Returns** `<NR1>`

**Examples** `DISP:WIND1:TRAC2:MEM:Y:SCAL:RPOS 3` sets 3 for the position of the reference division line on trace 2 for channel 1.

`DISP:WIND2:TRAC3:MEM:Y:SCAL:RPOS?` may return 4, which is the position of the reference division line for trace 3 on channel 2.

## DISPlay:CHANnel<x>:X:SPACing

Set or query the type of display of the graph horizontal axis on the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:X:SPACing { LINear | OBASe }
DISPlay:CHANnel<x>:X:SPACing?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`LINear` sets the display type to linear frequency for the graph horizontal axis, with the minimum frequency at the left edge and maximum frequency at the right edge.

`OBASe` sets the display type to order base for the graph horizontal axis, where measurement point numbers are positioned evenly in order of measurement.

**Returns** `LINear` means the frequency base (linear frequency axis with the minimum frequency at the left edge and the maximum frequency at the right edge).

`OBASe` specifies the order base (axis in which measurement point numbers are positioned evenly in the order of measurement).

**Examples** `DISP:WIND1:X:SPAC OBAS` sets order base as the display type of the graph horizontal axis for channel 1.

`DISP:WIND2:X:SPAC?` may return LINear, which is the display type of the graph horizontal axis on channel 2.

## DISPlay:CHANnel<x>:Y:[SCALe]:DIVisions

Set or query the number of divisions in all graphs on the specified channel.

**Group** Display commands

**Syntax**

```
DISPlay:CHANnel<x>:Y:[SCALe]:DIVisions <value>
DISPlay:CHANnel<x>:Y:[SCALe]:DIVisions?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of divisions per graph. Range: 4 to 30.

**Returns** `<NR1>`

**Examples** `DISP:WIND1:Y:SCAL:DIV 15` sets 15 as the number of divisions on graphs for channel 1.

`DISP:WIND1:Y:SCAL:DIV?` may return 4, which is the number of divisions on graphs set for channel 2.

## DISPlay:ENABle

Set or query the state of the display of the update on the graph area.

**Group** Display commands

**Syntax**

```
DISPlay:ENABle <value>
DISPlay:ENABle?
```

**Arguments** `<value>::=<boolean>`

**Returns** 0 means the update is hidden on the graph area.

1 means the update shows on the graph area.

**Examples** `DISP:ENAB 1` enables the display of the update on the graph area.

`DISP:ENAB?` may return 0, which means the display of the update is disabled for the graph area.

## DISPlay:FONT:[BAR]

Set or query the menu bar text font size.

**Group** Display commands

**Syntax**

```
DISPlay:FONT:[BAR] { SIZE }
DISPlay:FONT:[BAR]?
```

**Related Commands** `DISPlay:MENU:[STATe]`

**Arguments** `<size>::=<NR1>` Range: 8 to 20

**Returns** `<NR1>`

**Examples**

`DISP:FONT:BAR 10` sets 10 as the font size for the menu bar text.

`DISP:FONT:BAR?` may return 15, which is the font size for the menu bar text.

## DISPlay:FONT:GRAPh

Set or query the graph text font size.

**Group**

Display commands

**Syntax**

```
DISPlay:FONT:GRAPh <value>
DISPlay:FONT:GRAPh?
```

**Arguments**

`<value>::=<NR1>` Range: 8 to 20

**Returns**

`<NR1>`

**Examples**

`DISP:FONT:GRAP?` may return 11, which is font size of the graph text.

`DISP:FONT:GRAP 15` sets 15 to be the font size of the graph text.

## DISPlay:FSIGn

Set or query the display of the fail sign for the limit, ripple limit, and bandwidth tests.

**Group**

Display commands

**Syntax**

```
DISPlay:FSIGn <value>
DISPlay:FSIGn?
```

**Related Commands**

CALCulate<x>:[SELected]:LIMit:[STATe]

CALCulate<x>:[SELected]:RLIMit:[STATe]

CALCulate<x>:[SELected]:BLIMit:[STATe]

**Arguments**

`<value>::=<boolean>`

**Returns**

1 means the fail sign shows.

0 means the fail sign is hidden.

**Examples**

`DISP:FSIGn 1` enables the display of the fail sign for the limit test.

`DISP:FSI?` may return 0, which means the display of the fail sign is disabled for the ripple limit test.

## DISPlay:GRAPh<x>:[ACTivate] (No Query Form)

Set the specified graph as the active graph. The channel layout displays.

**Group**

Display commands

**Syntax**

```
DISPlay:GRAPh<x>:[ACTivate] <value>
```

**Related Commands**

DISPlay:GRAPh<x>:[ACTivate]:CHANnel

**Arguments**

**Command variables.** <x> is the graph number (1 to 16).

None

**Examples**

`DISP:GRAP2:ACT 2` assigns graph 2 as the active graph (if the channel layout is not set to D1, displaying only channel 1 or a single graph).

## DISPlay:GRAPh<x>:[ACTivate]:CHANnel (No Query Form)

Set the specified channel to the specified graph, only if the channel layout displays the corresponding specified graph value and the specified channel value is enabled.

**Group**

Display commands

**Syntax**

```
DISPlay:GRAPh<x>:[ACTivate]:CHANnel <value>
```

**Related Commands**

SERVice:CHANnel<x>:COUNt

**Arguments**

**Command variables.** <x> is the graph number (1 to 16).

**Examples** `DISP:GRAP2:ACT:CHAN 1` assigns channel 1 to graph 2. If there is only 1 active trace on channel 1, graph 2 will indicate the trace is not enabled.

## DISPlay:GRAPh<x>:[ACTivate]:LABel (No Query Form)

Set to display or hide the Y-axis label on the specified graph.

**Group** Display commands

**Syntax** `DISPlay:GRAPh<x>:[ACTivate]:LABel <value>`

**Arguments** **Command variables.** <x> is the graph number (1 to 16).

`<value>::=<boolean>`

**Examples** `DISP:GRAP2:ACT:LAB 1` enables the Y-axis label on graph 2.

`DISP:GRAP3:ACT:LAB 0` disables the Y-axis label on graph 3.

## DISPlay:GRAPh<x>:[ACTivate]:TITLe:DATA (No Query Form)

Set the title of the specified graph.

**Group** Display commands

**Syntax** `DISPlay:GRAPh<x>:[ACTivate]:TITLe:DATA <value>`

**Arguments** **Command variables.** <x> is the graph number (1 to 16).

`<value>::=<string>` is the title of the specified graph.

**Examples** `DISP:GRAP1:ACT:TITL:DATA "FILTER RESPONSE"` sets the title of graph 1 to Filter Response.

## DISPlay:GRAPh<x>:[ACTivate]:TITLe:[STATe] (No Query Form)

Set to display or hide the channel title for the specified graph.

**Group** Display commands

**Syntax**

```
DISPlay:GRAPh<x>:[ACTivate]:TITLe:[STATe] <value>
```

**Arguments**

**Command variables.** <x> is the graph number (1 to 16).

`<value>::=<boolean>`

**Examples**

`DISP:GRAP1:ACT:TITL:STAT 1` enables the display of the channel title for graph 1.

`DISP:GRAP2:ACT:TITL:STAT 0` disables the display of the channel title for graph 2.

## DISPlay:GRAPh<x>:[ACTivate]:TRACes

Set or query the specified traces to the specified graph for the active channel.

**Conditions**

Set the active graph first using `DISPlay:GRAPh<x>:[ACTivate]:CHANnel`.

**Group**

Display commands

**Syntax**

```
DISPlay:GRAPh<x>:[ACTivate]:TRACes <value>
DISPlay:GRAPh<x>:[ACTivate]:TRACes?
```

**Related Commands**

DISPlay:GRAPh<x>:[ACTivate]:CHANnel

**Arguments**

**Command variables.** <x> is the graph number (1 to 16).

`<value>::=<NR1>` is the trace(s) assigned to the graph in the format [numeric1], [numeric2], ... [numeric16]. Range: 1 to 16.

**Returns**

`<NR1>`

**Examples**

`DISP:GRAP2:ACT:CHAN 1` activates channel 1 to graph 2.

`DISP:GRAP2:ACT:TRAC 1,3,5,7,9,11,13,15` allocates these traces to graph 2 on channel 1.

## DISPlay:IMAGe

Set or query the color scheme of the display.

**Group** Display commands

**Syntax**

```
DISPlay:IMAGe { NORMal | INVert }
DISPlay:IMAGe?
```

**Arguments** `NORMal` is the normal color scheme (black background).

`INVert` is the inverted color scheme (white background).

**Returns** `NORMal` means a black background is set.

`INVert` means a white background is set.

**Examples** `DISP:IMAG NORM` sets the display to a normal color scheme.

`DISP:IMAG?` may return INVert, which means the display is set to an inverted color scheme.

## DISPlay:MAXimize

Set or query maximization for the active channel.

**Group** Display commands

**Syntax**

```
DISPlay:MAXimize <value>
DISPlay:MAXimize?
```

**Arguments** `<value>::=<boolean>`

**Returns** 0 means maximization is turned off.

1 means maximization is turned on.

**Examples** `DISP:MAX 1` enables maximization for the active channel.

`DISP:MAX?` may return 0, which means maximization is disabled for the active channel.

## DISPlay:MENU:[STATe]

Set or query the state of the display of the menu bar.

**Group** Display commands

**Syntax**

```
DISPlay:MENU:[STATe] <value>
DISPlay:MENU:[STATe]?
```

**Related Commands** DISPlay:FONT:[BAR]

**Arguments** `<value>::=<boolean>`

**Returns** 0 means the menu bar is hidden.

1 means the menu bar shows.

**Examples** `DISP:MENU:STAT 0` disables the display of the menu bar.

`DISP:MENU:STAT?` may return 1, which means the display of the menu bar is enabled.

## DISPlay:SPLit

Set or query the display layout for the channels.

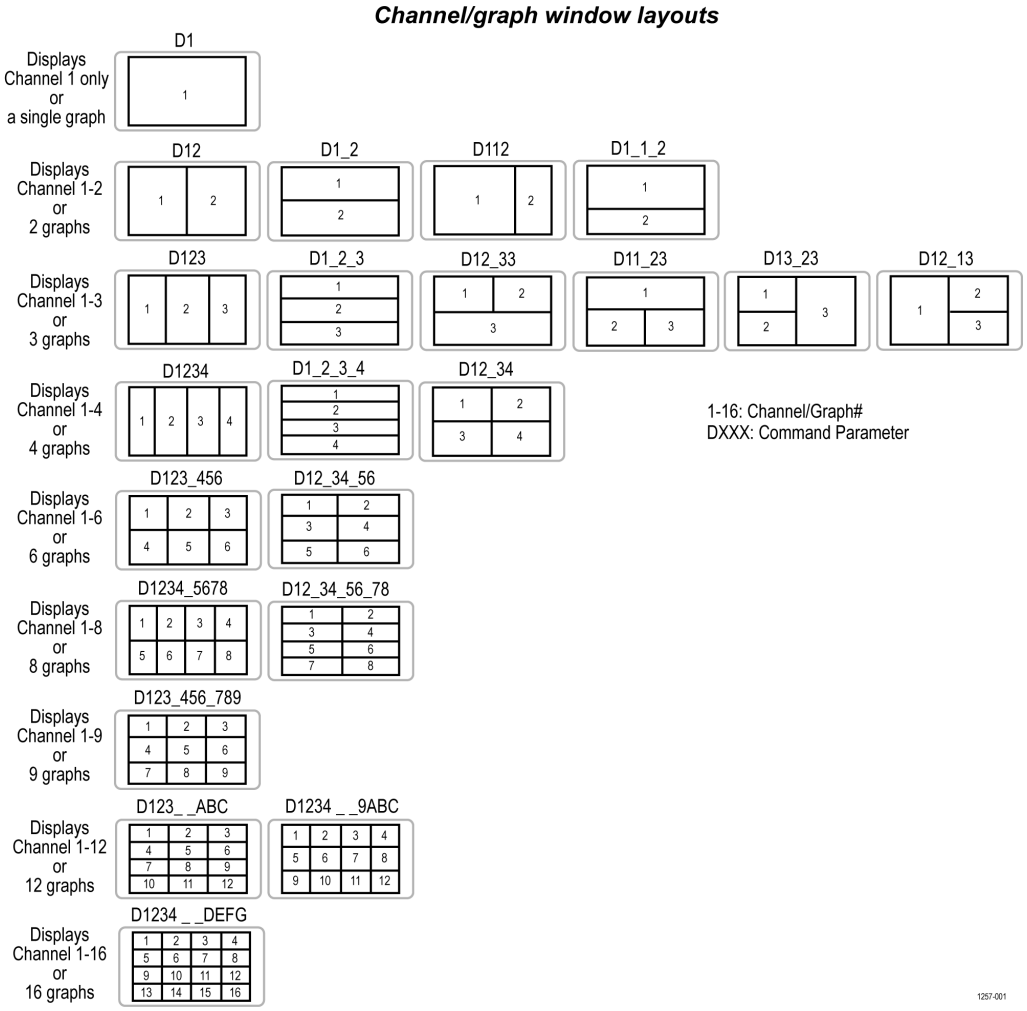
**Group** Display commands

**Syntax**

```
DISPlay:SPLit { D1 | D12 | D1_2 | D112 | D1_1_2 | D123
| D1_2_3 | D12_33 | D11_23 | D13_23 | D12_13 | D1234 |
D1_2_3_4 | D12_34 | D123_456 | D12_34_56 | D1234_5678 |
D12_34_56_78 | D123_456_789 | D123__ABC | D1234__9ABC |
D1234__DEFG }
DISPlay:SPLit?
```

**Related Commands** CALCulate<x>:PARameter:COUNt

**Arguments** Select one of these graph display layouts:




**Returns** See Arguments.

**Examples** `DISP:SPL D12_34` sets D12_34 as the layout for the channels.

`DISP:SPL?` may return D12_34, which is the layout set for the channels.

## DISPLay:TABLe:[STATe]

Set or query the status of the display window located on the lower part of the screen.

**Group** Display commands

**Syntax**

```
DISPLay:TABLe:[STATe] <value>
DISPLay:TABLe:[STATe]?
```

**Arguments**

```
<value>::=<boolean>
```

**Returns**

0 means the lower part of the screen is hidden.

1 means the lower part of the screen shows.

**Examples**

`DISP:TAB:STAT 0` turns off the display of the lower part of the screen.

`DISP:TAB:STAT?` may return 1, which means the display of the lower part of the screen is turned on.

## DISPlay:TABLe:TYPE

Set or query the type of window displaying on the lower part of the screen.

**Group**

Display commands

**Syntax**

```
DISPlay:TABLe:TYPE { MARKer | LIMit | SEGMent | ECHO | PLOSs
| SCFactor | RLIMit }
DISPlay:TABLe:TYPE?
```

**Arguments**

Select one of the following window types:

| Argument | Description |
|---|---|
| MARKer | Marker table window |
| LIMit | Limit test table window |
| SEGMent | Segment table window |
| ECHO | Echo window |
| PLOSs | Loss compensation table window |
| SCFactor | Power Sensor's calibration factor table window |
| RLIMit | Ripple test table window |

**Returns**

See Arguments.

**Examples**

`DISP:TABL:TYPE ECHO` sets echo, as the window type set for the lower part of the screen.

`DISP:TABL:TYPE?` may return MARK, which is the window type set for the lower part of the screen.

## DISPlay:UPDate:[IMMediate] (No Query Form)

Execute a single immediate update of the display when `DISPlay:ENABle` has set updates to 0.

**Group** Display commands

**Syntax** `DISPlay:UPDate:[IMMediate]`

**Related Commands** DISPlay:ENABle

**Arguments** None

**Examples** `DISP:UPD:IMM` updates the display once when `DISP:ENAB` is set to 0.

## DISPlay:WINDow<x>:[ACTivate] (No Query Form)

Set the specified channel as the active channel on the active graph. The channel must be enabled to allow the specified channel to be set as the active channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Display commands

**Syntax** `DISPlay:WINDow<x>:[ACTivate]`

**Examples** `DISP:WIND2:ACT` sets channel 2 as the active channel, if channel 2 is enabled.

## DISPlay:WINDow<x>:TRACe<y>:[STATe]

Set or query the state of the display for the data trace on specified trace for the specified channel.

**Conditions** Display commands

**Group** Display commands

**Syntax**

```
DISPlay:WINDow<x>:TRACe<y>:[STATe] <value>
DISPlay:WINDow<x>:TRACe<y>:[STATe]?
```

**Arguments** `<value>::=<boolean>` is the value representing the state of the display.

**Returns** 0 means the specified data trace is turned off.

1 means the specified data trace turned on.

**Examples** `DISP:WIND2:TRAC3:MEM:STAT 0` disables the display of data trace 3 for channel 2.

`DISP:WIND1:TRAC2:MEM:STAT?` may return 1, which means data trace 2 is disabled for channel 1.

## *ESE

Set or query the bits in the Event Status Enable Register (ESER). The ESER prevents events from being reported to the Status Byte Register (STB). Refer to Section 3, *Status and Events*, for the register information.

**Group** IEEE common commands

**Syntax**

```
*ESE <value>
*ESE?
```

**Related Commands** *WAI

*ESE

*SRE

*STB?

**Arguments** `<value>::=<NRf>` is the value in the range from 0 to 255.

***NOTE.*** *The binary bits of the ESER are set based on this value.*

**Returns** `<NRf>`

**Examples**

`*ESE 145` sets the ESER to binary 10010001, which enables PON, EXE, and OPC bits.

`*ESE?` may return the string **ESE 184*, which shows that ESER contains the binary value 10111000.

## *ESR? (Query Only)

Query the contents of the Standard Event Status Register (SESR). `*ESR?` also clears the SESR (since reading the SESR clears it). Refer to section 3, *Status and Events*, in this document for the register information.

**Group**

IEEE common commands

**Syntax**

`*ESR?`

**Related Commands**

*WAI

*ESE

*SRE

*STB?

**Returns**

`<NRf>`

**Examples**

`*ESR?` may return 213, which shows that SESR contains binary 11010101.

## *IDN? (Query Only)

Query the analyzer identification code.

**Group**

IEEE common commands

**Syntax**

`*IDN?`

**Returns**

The analyzer identification code in the following format:

`TEKTRONIX,TTR5xx,<serial_number>,<firmware_version>,`

where:

| Value | Description |
|---|---|
| TEKTRONIX | Indicates that Tektronix is the manufacturer. |
| TTR5xx | Indicates the model number. |
| <serial_number> | Indicates the serial number. |
| <firmware_version> | Indicates the firmware version. |

**Examples**

*IDN? may return TEKTRONIX, TTR503, B000111, FV1.3.2100, which correspond to the manufacturer name, model number, serial number, and firmware version.

## INITiate<x>:CONTinuous

Set or query the status of the trigger mode (continuous or hold) for the specified channel.

**Group**

Initiate commands

**Syntax**

```
INITiate<x>:CONTinuous <value>
INITiate<x>:CONTinuous?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

```
<value>::=<boolean>
```

**Returns**

0 means the trigger system is set to hold.

1 means the trigger system is set to continuous.

**Examples**

INIT1:CONT 1 sets the trigger mode to continuous for channel 1.

INIT2:CONT? may return 0, which means the continuous mode is disabled, setting trigger system to hold for channel 2.

## INITiate<x>:[IMMediate] (No Query Form)

Execute a single sweep for the specified channel, while the channel is being set to hold. After executing the measurement once, the trigger system for the channel returns to an idle state.

When you execute this command for a channel that is not idle (with hold or continuous mode enabled), an error occurs.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Initiate commands

**Syntax** `INITiate<x>:[IMMediate]`

**Related Commands** INITiate<x>:CONTinuous

**Arguments** None

**Examples** `INIT1:IMM` executes a single sweep for channel 1, while the triggering mode for channel 1 is being set to hold.

## MMEMory:LOAD:ASCFactor (No Query Form)

Load the power sensor calibration factor table for the specified port from the specified CSV file for the active channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:LOAD:ASCFactor <port>,<string>`

**Related Commands** SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:DATA

MMEMory:STORe:PSCFactor

**Arguments** `<port>::=<NR1>` is the port number (1 or 2) at which you are performing the calibration.

`<string>` is the CSV file name and destination of the calibration coefficient table. Range: 254 characters or less.

**Examples** `MMEM:LOAD:ASCF 1 “C:\Users\sensor1.csv”` loads the calibration factor table from the file *sensor1.csv* for port 1 on the active channel.

## MMEMory:LOAD:CKIT (No Query Form)

Load the definition of the specified calibration kit from the specified CSV file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:LOAD:CKIT <value>`

**Related Commands** MMEMory:STORe:CKIT

**Arguments** `<value>::=<string>` is the CSV file name and destination, which contains the definition of the calibration kit. Range: 254 characters or less.

**Examples** `MMEM:LOAD:CKIT 1 "C:\Users\filter1.csv"` loads the file *filter1.csv* to define the calibration kit 1.

## MMEMory:LOAD:LIMit (No Query Form)

Load the limit table from the specified CSV file for the active trace. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:LOAD:LIMit <value>`

**Related Commands** CALCulate<x>:[SELected]:LIMit:DATA

MMEMory:STORe:LIMit

**Arguments** `<value>::=<string>` is the CSV file name and destination, which contains the limit table. Range: 254 characters or less.

**Examples** `MMEM:LOAD:LIM "C:\Users\limitTable1.csv"` loads the limit table from the file *limitTable1.csv* for the active trace.

## MMEMory:LOAD:PROSs (No Query Form)

Load the information contained on the specified CSV file for the loss compensation table of the power calibration for the active channel on the specified port. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:LOAD:PROSs <value>`

**Related Commands** SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:ASENsor:DATA

MMEMory:STORe:PLOSs

**Arguments** `<value>::=<string>` is the CSV file name and destination, which contains the loss compensation table for power calibration. Range: 254 characters or less.

**Examples** `MMEM:LOAD:PROS 1 "C:\Users\lc.csv"` loads the file *lc.csv* to define the loss compensation table for the power calibration for port 1 on the active channel.

## MMEMory:LOAD:RLIMit (No Query Form)

Load the ripple limit table from the specified CSV file to the active trace. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:LOAD:RLIMit <value>`

**Related Commands** CALCulate<x>:[SELected]:RLIMit:DATAMMEMory:STORe:RLIMit

**Arguments** `<value>::=<string>` is the CSV file name and destination for the ripple limit table. Range: 254 characters or less.

**Examples** `MMEM:LOAD:RLIM "C:\Users\rippleTable1.csv"` loads the ripple limit table from the file *rippleTable1.csv* for the active trace.

## MMEMory:LOAD:SEGMent (No Query Form)

Load the segment sweep table for the active channel from the specified CSV file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax**

```
MMEMory:LOAD:SEGMent <value>
```

**Related Commands** SENSe<x>:SEGMent:DATA

MMEMory:STORe:SEGMent

**Arguments** `<value>::=<string>` is the CSV file name and destination for the segment sweep table. Range: 254 characters or less.

**Examples** `MMEM:LOAD:SEGM "C:\Users\segmentTable1.csv"` loads the segment limit table from the file *segmentTable1.csv* for the active channel.

## MMEMory:LOAD:[STATe] (No Query Form)

Load the specified instrument state file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax**

```
MMEMory:LOAD:[STATe] <value>
```

**Arguments** `<value>::=<string>` is the state CSV file name, extension, and destination, where the extension and content saved are described in the following table:

| Argument | Description |
|---|---|
| state | Contains measurement conditions. |
| cstate | Contains measurement conditions and calibration data. |
| dstate | Contains measurement conditions and trace data. |
| cdstate | Contains measure conditions, calibration state, and trace data. |

Range: 254 characters or less

**Examples** `MMEM:LOAD:STAT "C:\Users\inst1.cdstate"` loads the file *inst1.xml* to define the state of the instrument.

## MMEMory:STORe:CKIT (No Query Form)

Save the definition of the specified calibration kit to the specified CSV file on the active channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:STORe:CKIT <file>`

**Related Commands** MMEMory:LOAD:CKIT

**Arguments** **Command variables.** <x> is the calibration kit number (1 to 20).

`<file>::=<string>` is the CSV file name and destination, which contains the definition of the specified calibration kit. Range: 254 characters or less.

`<value>::=<NR1>` is the calibration kit (1 or 2).

**Examples** `MMEM:STOR:CKIT 2 "C:\Users\filter1.csv"` saves the definition of the specified calibration kit from the file *filter1.csv* for port 2 on the active channel.

## MMEMory:STORe:LIMit (No Query Form)

Save the limit table to the specified CSV file on the active trace. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax** `MMEMory:STORe:LIMit <value>`

**Related Commands** CALCulate<x>:[SELected]:LIMit:DATA

MMEMory:LOAD:LIMit

**Arguments** `<value>::=<string>` is the CSV file name and destination, which contains the user defined limit table. Range: 254 characters or less.

**Examples** `MMEM:STOR:LIM ”C:\Users\limitTable1.csv”` saves the limit table to the file *limitTable1.csv* for the active trace.

## MMEMory:STORe:PLOSs (No Query Form)

Save the loss compensation table for power calibration specified in the CSV file for the specified port and channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax**

```
MMEMory:STORe:PLOSs <port>,<value>
```

**Related Commands** SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:ASENsor:DATA

MMEMory:LOAD:PROSs

**Arguments** `<port>::=<NR1>` is the port number for which you saved the loss compensation table for the power calibration.

`<value>::=<string>` is the CSV file name and destination containing the loss compensation table for the power calibration. Range: 254 characters or less.

**Examples** `MMEM:STOR:PLOS 1 “C:\Users\lctable.csv”` saves the loss compensation table for power calibration to the file *lctable.csv.* for port 1 for the active channel.

## MMEMory:STORe:PSCFactor (No Query Form)

Save the power sensor calibration factor table for the specified port to the specified CSV file for the active channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax**

```
MMEMory:STORe:PSCFactor <port>,<string>
MMEMory:STORe:PSCFactor?
```

**Related Commands**

SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:DATA

MMEMory:LOAD:ASCFactor

**Arguments**

`<port>::=<NR1>` is the port number (1 or 2).

`<string>` is the CSV file name and destination, which contains the power sensor calibration factor table. Range: 254 characters or less.

**Examples**

`MMEM:STOR:PSCF 1 "C:\Users\psfc.csv"` saves the power sensor calibration factor table from the file *psfc.csv* for port 1 on the active channel.

## MMEMory:STORe:RLIMit (No Query Form)

Save the ripple limit table for the active trace on the specified CSV file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group**

Memory commands

**Syntax**

`MMEMory:STORe:RLIMit <value>`

**Related Commands**

CALCulate<x>:[SELected]:LIMit:DATA

MMEMory:LOAD:RLIMit

**Arguments**

`<value>::=<string>` is the CSV file name and destination, which contains the user defined ripple limit table. Range: 254 characters or less.

**Examples**

`MMEM:STOR:RLIM "rippleTable1.csv"` saves the ripple limit table to the file *rippleTable1.csv* for the active trace.

## MMEMory:STORe:SALL

Set or query what content (all channels/traces or only displayed channels/traces) must be saved as instrument state. Make sure the CSV is available at the provided file path as no notification is provided if the file is not loaded.

**Group**

Memory commands

**Syntax**

```
MMEMory:STORe:SALL <value>
MMEMory:STORe:SALL?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns**

1 means all channels and traces must be saved.

0 means only displayed channels and traces must be saved.

**Examples**

`MMEM:STOR:SALL 1` saves all channels and traces as instrument state.

`MMEM:STOR:SALL?` may return 0, which means only displayed channels and traces must be saved as instrument state.

## MMEMory:STORe:SEGMent (No Query Form)

Store the segment sweep table for the active channel from the specified CSV file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group**

Memory commands

**Syntax**

```
MMEMory:STORe:SEGMent <value>
```

**Related Commands**

SENSe<x>:SEGMent:DATA

MMEMory:LOAD:SEGMent

**Arguments**

`<value>::=<string>` is the CSV file name and destination, which contains the user defined segment table. Range: 254 characters or less.

**Examples**

`MMEM:STOR:SEGM "C:\Users\rippleTable1.csv"` stores the segment limit table to the file *rippleTable1.csv* for the active channel.

## MMEMory:STORe:SNP:[DATA] (No Query Form)

Save the measurement data as a touchstone file in SnP format (S1P or S2P). The measurement data format is applied by `MMEMory:STORe:SNP:FORMat`, while the SnP format is set by `MMEMory:STORe:SNP:TYPE:S1P` or

`MMEMory:STORe:SNP:TYPE:S2P`. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

***NOTE.*** *The SnP file format is not automatically appended to the file name.*

**Group** Memory commands

**Syntax** `MMEMory:STORe:SNP:[DATA] <value>`

**Related Commands**

MMEMory:STORe:SNP:TYPE:S1P

MMEMory:STORe:SNP:TYPE:S2P

MMEMory:STORe:SNP:FORMat

MMEMory:STORe:SNP:PORTS?

**Arguments** `<value>::=<string>` is the CSV file name and destination in SnP format.

**Examples** In the following example, command 1 species the SnP touchstone format as S2P for ports 1 and 2. Command 2 sets the data format (used to save measurement data in touchstone files) to Logarithmic Magnitude with Angle. Command 3 saves the measurement data as a touchstone file in the SnP format (*,test.s2p*), and stores the saved measurement data in the directory *C:\Users*:

1. `MMEM:STOR:SNP:TYPE:S2P 1,2`
2. `MMEM:STOR:SNP:FORM MA`
3. `MMEM:STOR:SNP:DATA "C:\Users\test.s2p"`

## MMEMory:STORe:SNP:FORMat

Set or query the data format used to save measurement data in a touchstone file for the active channel. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

***NOTE.*** *The SnP file format is not automatically appended to the file name.*

**Group** Memory commands

**Syntax**

```
MMEMory:STORe:SNP:FORMat { AUTO | MA | DB | RI }
MMEMory:STORe:SNP:FORMat?
```

**Arguments**

| Argument | Description |
|---|---|
| AUTO | Sets data format to DB. |
| MA | Sets data format to log magnitude + angle. |
| DB | Sets data format to linear magnitude + angle. |
| RI | Sets data format to real + imaginary. |

**Returns**

See Arguments.

**Examples**

`MMEM:STOR:SNP:FORM DB` sets the data format to Log Magnitude + angle for the active channel.

`MMEM:STOR:SNP:FORM?` may return RI, which is the data format real + imaginary for the active channel.

## MMEMory:STORe:SNP:NPORTS? (Query Only)

Query the number of ports used when saving a touchstone file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

***NOTE.*** *The SnP file format is not automatically appended to the file name.*

**Group**

Memory commands

**Syntax**

```
MMEMory:STORe:SNP:NPORTS?
```

**Returns**

Number of ports used when saving a touchstone file in the SnP format

**Examples**

`MMEM:STOR:SNP:NPORTS?` may return 1, which indicates the number of ports used to save the touchstone file. The format used to save the touchstone file was S1P.

## MMEMory:STORe:SNP:PORTS? (Query Only)

Query the specific ports used when saving a touchstone file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

***NOTE.*** *The SnP file format is not automatically appended to the file name.*

**Group** Memory commands

**Syntax** `MMEMory:STORe:SNP:PORTS?`

**Returns** 1 means port 1 was used.

2 means port 2 was used.

**Examples** `MMEM:STOR:SNP:PORTS?` may return 1, 2 as the specific ports used, signifying the touchstone file is saved in S2P format.

## MMEMory:STORe:SNP:TYPE:S1P (No Query Form)

Set the port used for the S1P file type when saving measurement data for the active channel to touchstone format. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded.

***NOTE.*** *The SnP file format is not automatically appended to the file name.*

**Group** Memory commands

**Syntax** `MMEMory:STORe:SNP:TYPE:S1P <value>`

**Related Commands** MMEMory:STORe:SNP:NPORTS?

MMEMory:STORe:SNP:[DATA]

**Arguments** `<value>::=<NR1>` is the port number (1 or 2) used when saving a touchstone file in S1P format.

**Examples** `MMEM:STOR:SNP:TYPE:S1P 1` sets port 1 as the desired port to be saved in the touchstone file.

## MMEMory:STORe:SNP:TYPE:S2P (No Query Form)

Set the ports used for the S2P file type when saving measurement data to touchstone format for the active channel. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded.

***NOTE.*** *The SnP file format is not automatically appended to the file name.*

**Group** Memory commands

**Syntax**

```
MMEMory:STORe:SNP:TYPE:S2P <value1>,<value2>
```

**Related Commands** MMEMory:STORe:SNP:NPORTS?

MMEMory:STORe:SNP:[DATA]

**Arguments** `<value1>::=<NR1>` is the first port number (1 or 2) used for the S2P format.

`<value2>::=<NR1>` is the second port number (1 or 2) used for the S2P format.

**Examples** `MMEM:STOR:SNP:TYPE:S1P 1,2` sets port 1 and 2 as the desired ports to be saved in the touchstone file.

## MMEMory:STORe:[STATe] (No Query Form)

Save the state of the instrument to the specified state file. Make sure the CSV file exists at the provided file path as no notification is provided if the file is not loaded.

**Group** Memory commands

**Syntax**

```
MMEMory:STORe:[STATe] <file>
```

**Arguments** `<file>::=<string>` is the CSV file name, extension, and destination used to save the desired state of the instrument, where the extension and the content saved is described in the following table:

| Argument | Description |
|---|---|
| state | Saves the measurement conditions only. |
| cstate | Saves the measurement conditions and calibration data. |
| dstate | Saves the measurement conditions and trace data. |
| cdstate | Saves measure conditions, calibration state, and trace data. |

**Examples**

`MMEM:STOR:STAT "C:\Users\instrument1.cdstate"` saves the instrument measurement conditions, calibration state, and trace data to the file *instrument1.cdstate*.

## MMEMory:STORe:STYPe? (Query Only)

Query the last extension used to save the instrument's state file. Make sure the CSV file is available at the provided file path as no notification is provided if the file is not loaded.

**Group**

Memory commands

**Syntax**

```
MMEMory:STORe:STYPe?  { STATe | CSTATe | DSTATe | CDSTate }
```

**Returns**

| Value | Description |
|---|---|
| STATe | Means only measurement conditions are saved. |
| CSTATe | Means measurement conditions and calibration state are saved. |
| DSTATe | Means measurement conditions and trace data are saved. |
| CDSTATe | Means measurement conditions, calibration state and trace data are saved. |

**Examples**

`MMEM:STOR:STYPE?` may return DSTATe, which indicates the last saved instrument sate file saved contained a dstate extension.

## *OPC

Generates the operation complete message in the Standard Event Status Register (SESR) when all pending operations are complete.

`*OPC?` places the ASCII character *1* into the output queue when all pending operations are complete.

The return from `*OPC?` will not be available for reading until all pending operations are complete.

`*OPC` synchronizes the operation of the analyzer with your application program.

***NOTE.*** *Refer to Synchronizing Execution , page 3-9 for details.*

**Conditions** Measurement views: All

**Group** IEEE common commands

**Syntax**

```
*OPC
*OPC?
```

**Arguments** None

**Returns** The operation complete message in SESR.

## OUTPUT:[STATe]

Set or query the state of the stimulus signal output. You can only make measurements when the signal output is enabled.

**Group** Output commands

**Syntax**

```
OUTPUT:[STATe] <value>
OUTPUT:[STATe]?
```

**Arguments** `<value>::=<boolean>`

**Returns** 1 means the stimulus signal output shows.

0 means the stimulus signal output is hidden.

**Examples** `OUTP:STAT 1` enables the stimulus signal output.

`OUTP:STAT?` may return 0, which means the stimulus signal output is disabled.

## *RST (No Query Form)

Query the instrument settings to the factory defaults. `*RST` does not alter:

- Alignment data that affects device specifications
- The output queue
- The service request enable register setting
- The standard event status enable register setting
- The power-on status clear flag setting
- Stored settings

***NOTE.*** *The execution of `*RST` is not complete until all changes from resetting the instrument are completed. Following commands and queries will not be executed until these actions are finalized.*

**Group** IEEE common commands

**Syntax** `*RST`

**Related Commands** *WAI

**Arguments** None

**Examples** `*RST` resets the instrument settings to the factory defaults.

## SENSe<x>:AVERage:CLEar (No Query Form)

Reset the data count to 0 when using the averaging function for the specified channel, which is set by `SENSe<x>:AVERage:[STATe]`. Measurement data recorded before executing this command is not used for averaging.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Sense commands

**Syntax** `SENSe<x>:AVERage:CLEar`

**Related Commands** SENSe<x>:AVERage:[STATe]

**Examples** `SENS1:AVER:CLE` resets the data count to 0, when using the averaging function for channel 1.

## SENSe<x>:AVERage:COUNt

Set or query the averaging factor value (number of traces to average) for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:AVERage:COUNt <value>
SENSe<x>:AVERage:COUNt?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the averaging factor. Range: 1 to 999.

**Returns** `<NR1>`

**Examples** `SENS1:AVER:COUN 400` sets 400 as the averaging factor for channel 1.

`SENS2:AVER:COUN?` may return 999, which is the averaging factor set for channel 2.

## SENSe<x>:AVERage:[STATe]

Set or query the state of the averaging function for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:AVERage:[STATe] <value>
SENSe<x>:AVERage:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the averaging function is turned on.

0 means the averaging function is turned off.

**Examples**

`SENS1:AVER:STAT 1` enables the averaging function for channel 1.

`SENS2:AVER:STAT?` may return 0, which means the averaging function is disabled for channel 2.

## SENSe<x>:BANDwidth:RESolution

Set or query the IF bandwidth value for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:BANDwidth:RESolution <value>
SENSe<x>:BANDwidth:RESolution?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the IF bandwidth. Range: 1 to 5e5 Hz.

**Returns**

`<NRf>`

**Examples**

`SENS1:BAND:RES 5e5` sets 0.5 MHz as the value of the IF bandwidth for channel 1.

`SENS2:BAND:RES?` may return 4e5, which means 0.4 MHz is the IF bandwidth set for channel 2.

## SENSe<x>:CORRection:CLEar (No Query Form)

Clear the calibration error coefficients for the specified channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:CLEar
```

**Arguments**

None

**Examples** `SENS1:CORR:CLE` clears the calibration error coefficients for channel 1.

## SENSe<x>:CORRection:COEFficient:[DATA]

Set or query the calibration coefficient data for the specified channel. When the calibration factor is interpolated, the interpolated calibration coefficient array is read. The written value in the coefficient array becomes effective only after you save the coefficient array using `SENSe<x>:CORRection:COEFficient:SAVE`.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:[DATA] {
<string>,<value1>,<value2> } <array>
SENSe<x>:CORRection:COEFficient:[DATA]?
```

**Related Commands**

SENSe<x>:CORRection:COEFficient:METHod:ERESPonse

SENSe<x>:CORRection:COEFficient:SAVE

SENSe<x>:CORRection:COEFficient:METHod:RESPonse:OPEN

SENSe<x>:CORRection:COEFficient:METHod:RESPonse:SHORt

SENSe<x>:CORRection:COEFficient:METHod:RESPonse:THRU

SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COEFficient:METHod:SOLT1

SENSe<x>:CORRection:COEFficient:METHod:SOLT2

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<string>` is the calibration type:

| String | Description |
|---|---|
| ES | Source match |
| ER | Reflection tracking |
| ED | Directivity |
| EL | Load match |
| ET | Transmission tracking |
| EX | Isolation |

`<value1>::=<NR1>` is the response port (1 or 2).

`<value2>::=<NR1>` is the stimulus port (1 or 2).

`<array>::=<NRf>` is the calibration coefficient array of the number of measurement points NP*2. For any value *n*, being the integer value between 1 and NP:

`Data(n*2-1)` is the data (real value) at the $n^{th}$ measurement point.

`Data(n*2)` is the data (imaginary value) at the $n^{th}$ measurement point.

**Returns** `<string>`

**Examples** `SENS1:CORR:COEF:DATA ES, 1,1,2.54,2.4,0.86,-6.7,0.71,-0.97` sets the calibration coefficient array to source match for port 1 on channel 1.

`SENS2:CORR:COEF:DATA? ES, 1,1` may return 2.54622, 2.38533, 0.859613, -6.70929, 0.709504, -0.96708, as the source match calibration coefficient array for channel 2.

## SENSe<x>:CORRection:COEFficient:HDR

Set or query the HDR gain path for each port on the specified channel for error term purposes.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:HDR <value>
SENSe<x>:CORRection:COEFficient:HDR?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 0 means the gain path is set to low.

1 means the gain path is set to high.

**Examples** `SENS1:CORR:COEF:HDR 1` sets the gain path to high for each port on channel 1.

`SENS4:CORR:COEF:HDR?` may return 0, which means the gain path is set to low for each port on channel 4.

## SENSe<x>:CORRection:COEFficient:METHod:ERESPonse (No Query Form)

Set the calibration type to (enhanced) response calibration for the specified channel and port.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:METHod:ERESPonse
<value1>,<value2>
```

***NOTE.*** *You must specify a different port number for each parameter.*

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value1>::=<NR1>` is the response port (1 or 2).

`<value2>::=<NR1>` is the stimulus port (1 or 2).

**Examples** `SENS1:CORR:COEF:METH:ERES 1,2` sets the (enhanced) response calibration to channel 1, assigning port 1 as the response port and port 2 as the stimulus port.

## SENSe<x>:CORRection:COEFficient:METHod:RESPonse:OPEN (No Query Form)

Set the calibration type to response calibration (open) for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:METHod:RESPonse:OPEN <value>
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the response port (1 or 2).

**Examples** `SENS1:CORR:COEF:METH:RESP:OPEN 1` sets the response calibration (open) for channel 1, with port 1 as the response port.

## SENSe<x>:CORRection:COEFficient:METHod:RESPonse:SHORt (No Query Form)

Set the calibration type to response calibration (short) for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:METHod:RESPonse:SHORt
<value>
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the response port (1 or 2).

**Examples** `SENS1:CORR:COEF:METH:RESP:SHOR 1` sets the response calibration (short) for channel 1 with port 1 as the response port.

## SENSe<x>:CORRection:COEFficient:METHod:RESPonse:THRU (No Query Form)

Set the response calibration type to response calibration (thru) for the specified input and output ports.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:METHod:RESPonse:THRU
<value1>,<value2>
```

***NOTE.*** *You must specify a different port number for each parameter.*

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value1>::=<NR1>` is the input port (1 or 2).

`<value2>::=<NR1>` is the output port (1 or 2).

**Examples** `SENS1:CORR:COEF:METH:RESP:THRU 2,1` sets the response calibration (thru) for channel 1, assigning port 2 as the input port and port 1 as the output port.

## SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS (No Query Form)

Set the number of ports used for the SOLT calibration procedure for the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS <value>`

**Related Commands** SENSe<x>:CORRection:COLLect:METHod:SOLT:PORTS

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of ports for the SOLT calibration (1 or 2).

**Examples** `SENS1:CORR:COEF:METH:SOLT:NPORTS 2` sets 2 as the number of ports to be used in the SOLT calibration for channel 1.

## SENSe<x>;:CORRection:COEFficient:METHod:SOLT:PORTS (No Query Form)

Sets the ports to use the SOLT calibration based on the number of ports set by `SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS` on the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>;:CORRection:COEFficient:METHod:SOLT:PORTS <array>`

**Arguments** `<array>` is the array of ports used for SOLT calibration, where the number of ports is set by `SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS`.

**Examples** `SENS1:CORR:COEF:METH:SOLT:NPORTS 2` sets 2 as the number of ports used for the SOLT calibration for channel 1.

`SENS1:CORR:COEF:METH:SOLT:PORTS 1,2` sets ports 1 and 2 to be used on the 2–port SOLT calibration for channel 1. This is equivalent to`SENS1:CORR:COEFt:METH:SOLT2`.

## SENSe<x>:CORRection:COEFficient:METHod:SOLT1 (No Query Form)

Set the calibration type to full 1-port calibration for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:METHod:SOLT1 <value>
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port (1 or 2), where the 1-port calibration is performed.

**Examples** `SENS1:CORR:COEF:METH:SOLT1 1` sets a full 1-port calibration for port 1 on channel 1.

## SENSe<x>:CORRection:COEFficient:METHod:SOLT2 (No Query Form)

Set the calibration type to full 2-port calibration for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:METHod:SOLT2
<value1>,<value2>
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value1>::=<NR1>` is the first port of the full 2-port calibration.

`<value2>::=<NR1>` is the second port of the full 2-port calibration.

**Examples** `SENS1:CORR:COEF:METH:SOLT2 1,2` sets a full 2-port calibration for ports 1 and 2 on channel 1.

## SENSe<x>:CORRection:COEFficient:SAVE (No Query Form)

Calculate and apply calibration coefficients depending on the calibration type selected when writing calibration data.

When you enable the calibration coefficients, this action clears all calibration data regardless of whether it was used for calculation. This action also clears the calibration type selection.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COEFficient:SAVE
```

**Related Commands**

SENSe<x>:CORRection:COEFficient:[DATA]

SENSe<x>:CORRection:COEFficient:METHod:ERESPonse

SENSe<x>:CORRection:COEFficient:METHod:RESPonse:OPEN

SENSe<x>:CORRection:COEFficient:METHod:RESPonse:SHORt

SENSe<x>:CORRection:COEFficient:METHod:RESPonse:THRU

SENSe<x>:CORRection:COEFficient:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COEFficient:METHod:SOLT1

SENSe<x>:CORRection:COEFficient:METHod:SOLT2

**Arguments** None

**Examples** `SENS1:CORR:COEF:SAVE` enables calibration coefficients after writing calibration data to channel 1.

## SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation (No Query Form)

Measure the port isolation calibration data from the specified stimulus port to the specified response port for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation
<value1>,<value2>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:METHod:ERESPonse

SENSe<x>:CORRection:COLLect:METHod:RESPonse:THRU

SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COLLect:METHod:SOLT2

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value1>::=<NR1>` is the response port (1 or 2).

`<value2>::=<NR1>` is the stimulus port (1 or 2).

**Examples** `SENS1:CORR:COLL:ACQ:ISO 1,2` measures the isolation calibration data from stimulus port 2 to response port 1 for channel 1.

## SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD (No Query Form)

Measure the calibration data of the load standard of the specified port for the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD <value>`

**Related Commands** SENSe<x>:CORRection:COLLect:METHod:ERESPonse

SENSe<x>:CORRection:COLLect:METHod:RESPonse:OPEN

SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt

SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COLLect:METHod:SOLT1

SENSe<x>:CORRection:COLLect:METHod:SOLT2

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port (1 or 2) where the load standard is calibrated.

**Examples** `SENS1:CORR:COLL:ACQ:LOAD 1` measures the calibration data for the load standard for port 1 on channel 1.

## SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN (No Query Form)

Measure the calibration data of the open standard for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN <value>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:METHod:ERESPonse

SENSe<x>:CORRection:COLLect:METHod:RESPonse:OPEN

SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COLLect:METHod:SOLT1

SENSe<x>:CORRection:COLLect:METHod:SOLT2

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port (1 or 2) where the open standard is calibrated.

**Examples**

`SENS1:CORR:COLL:ACQ:OPEN 1` measures the calibration data for the open standard on port 1 for channel 1.

## SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt (No Query Form)

Measure the calibration data of the short standard for the specified port on the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt <value>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:METHod:ERESPonse

SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt

SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COLLect:METHod:SOLT1

SENSe<x>:CORRection:COLLect:METHod:SOLT2

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port (1 or 2) where the short standard is calibrated.

**Examples**

`SENS1:CORR:COLL:ACQ:SHORT 1` measures the calibration data for the short standard for port 1 on channel 1.

## SENSe<x>:CORRection:COLLect:[ACQuire]:SUBClass

Set or query the subclass used for the calibration on the specified channel. For example, if you set two subclasses (thru1 and thru2), visible in the calibration menu, this command can select either (thru1 or thru2) to perform the calibration.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:[ACQuire]:SUBClass <value>
SENSe<x>:CORRection:COLLect:[ACQuire]:SUBClass?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of the standard subclass for the calibration. Range: 1 to 8.

**Returns** `<NR1>`

**Examples** `SENS1:CORR:COLL:ACQ:SUBC 4` sets 4 as the standard subclass for calibrations on channel 1.

`SENS2:CORR:COLL:ACQ:SUBC?` may return 3, which is the standard subclass 3 set for calibrations on channel 2.

## SENSe<x>:CORRection:COLLect:[ACQuire]:THRU (No Query Form)

Measure the calibration data of the thru standard from the specified stimulus port to the specified response port for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:[ACQuire]:THRU <value1>,<value2>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:METHod:ERESPonse

SENSe<x>:CORRection:COLLect:METHod:RESPonse:THRU

SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS

SENSe<x>:CORRection:COLLect:METHod:SOLT2

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value1>::=<NR1>` is the response port (1 or 2).

`<value2>::=<NR1>` is the stimulus port (1 or 2).

**Examples** `SENS3:CORR:COLL:ACQ:THRU 1,2` measures the calibration data of the thru standard from stimulus port 2 to response port 1 for channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:LABel

Set or query the name of the active calibration kit for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:LABel <value>
SENSe<x>:CORRection:COLLect:CKIT:LABel?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<string>` is the name of the active calibration kit. Range: 254 characters or less.

**Returns** `<string>`

**Examples** `SENS1:CORR:COLL:CKIT:LAB CKIT1` sets the name of the active calibration kit for channel 1 to CKIT1.

`SENS2:CORR:COLL:CKIT:LAB?` may return CEM, which is the name of the active calibration kit used for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:ORDer:LABel

Set or query the name of the subclass label selected for the active calibration kit for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:ORDer:LABel <value>
SENSe<x>:CORRection:COLLect:CKIT:ORDer:LABel?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<string>` is the subclass name of the calibration kit. Range: 254 characters or less.

**Returns**

`<string>`

**Examples**

`SENS1:CORR:COLL:CKIT:ORD:LAB SC1` sets SC1 as the subclass name for the calibration kit on channel 1.

`SENS2:CORR:COLL:CKIT:ORD:LAB?` may return SCIII, which is the name of the subclass used for the calibration kit on channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:ORDer:LOAD

Set or query the standard used for the load measurement for the active calibration kit on the specified port for the specified channel. You must set the specified standard type to load SENSe<x>:CORRection:COLLect:CKIT:STAN:TYPE before performing a calibration.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:ORDer:LOAD <value>,<port>
SENSe<x>:CORRection:COLLect:CKIT:ORDer:LOAD?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the standard number. Range: 1 to 21.

`<port>::=<NR1>` is the port number (1 or 2).

**Returns**

`<value>,<port>`

**Examples**

`SENS1:CORR:COLL:CKIT:ORD:LOAD 2,4` sets standard 4 as the load measurement for port 2 on channel 1.

`SENS2:CORR:COLL:CKIT:ORD:LOAD?` may return 4, 1, where standard 1 is the load measurement used for port 1 on channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:ORDer:OPEN

Set or query standard used for open measurement for the specified calibration kit on the specified port and channel.

You must set the specified standard type, using `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE`, before performing a calibration.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:ORDer:OPEN
<value1>,<value2>
SENSe<x>:CORRection:COLLect:CKIT:ORDer:OPEN?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value1>::=<NR1>` is the port number (1 or 2).

`<value2>::=<NR1>` is the number of the standard used for the open measurement. Range: 1 to 21.

**Returns**

`<value1>,<value2>`

**Examples**

`SENS1:CORR:COLL:CKIT:ORD:OPEN 2,4` sets standard 4 as the open measurement on port 2 for channel 1.

`SENS2:CORR:COLL:CKIT:ORD:OPEN?` may return 1, 3, where standard 3 is the open measurement used on port 1 for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

Set or query the subclass for the active calibration from the subclass menu for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect] <value>
SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]?
```

***NOTE.*** *You must specify a different port number for each parameter.*

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of the standard subclass for the active calibration. Range: 1 to 8.

**Returns** `<NR1>`

**Examples** `SENS1:CORR:COLL:CKIT:ORD:SEL 5` sets standard subclass 5 for calibrations on channel 1.

`SENS3:CORR:COLL:CKIT:ORD:SEL?` may return 2, which means standard subclass 2 is used for calibrations on channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:ORDer:SHORt

Set or query standard used for the short measurement for the active calibration kit on the specified port and channel, Before performing the calibration, you must set the specified standard type to short with `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE`.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:ORDer:SHORt <value>
SENSe<x>:CORRection:COLLect:CKIT:ORDer:SHORt?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of the standard. Range: 1 to 21.

**Returns** `<NR1>`

**Examples** `SENS1:CORR:COLL:CKIT:ORD:SHOR 2,4` sets standard 4 as the short measurement on port 2 for channel 1.

`SENS2:CORR:COLL:CKIT:ORD:SHOR?` may return 1, 3, where standard 3 is the short measurement used on port 1 for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:ORDer:THRU

Set or query the calibration kit standard selected for the thru measurement between the specified ports on the specified channel. You must set the specified standard type to thru or unknown thru, by using `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE` before performing a calibration.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:ORDer:THRU
<value>,<port1>,<port2>
SENSe<x>:CORRection:COLLect:CKIT:ORDer:THRU?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the standard calibration kit used for the thru measurement between the specified ports (input and output). Range: 1 to 21.

`<port1>::=<NR1>` is the input port (1 or 2).

`<port2>::=<NR1>` is the output port (1 or 2).

**Returns** `<value>,<port1>,<port2>`

**Examples** `SENS1:CORR:COLL:CKIT:ORD:THRU 2,1,6` sets standard 6 for thru measurement between ports 1 and 2 for calibrations on channel 1.

`SENS3:CORR:COLL:CKIT:ORD:THRU?` may return 1, 2, 4, where standard 4 is the thru measurement between ports 1 and 2 used for calibrations on channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:RESet (No Query Form)

Reset the standard for the calibration kit on the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:RESet
```

**Arguments** None

**Examples** `SENS3:CORR:COLL:CKIT:RES` resets the standard.

## SENSe<x>:CORRection:COLLect:CKIT:[SELect]

Set or query the index of the calibration kit for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:[SELect] <value>
SENSe<x>:CORRection:COLLect:CKIT:[SELect]?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:ORDer:[SELect]

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the standard number of the calibration kit. Range: 1 to 20.

**Returns** `<NR1>`

**Examples**

`SENS1:CORR:COLL:CKIT:SEL 3` sets index 3 for the calibration kit standard for channel 1.

`SENS2:CORR:COLL:CKIT:SEL?` may return 1, which means index 2 is set for the calibration kit standard for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:ARBitrary

Set or query the arbitrary impedance value of the arbitrary standard type of the specified calibration kit standard for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:ARBitrary <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:ARBitrary?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit. Range: 1 to 20.

`<value>::=<NRf>` is the arbitrary impedance value for the arbitrary standard type of the specified calibration kit standard. Range: -1e18 to 1e18 ; Units:Ω(omega).

If the specified variable is outside the allowable setup range, the minimum value or the maximum value is set.

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN3:ARB 100.4` sets 100.4 Ω as the value of the arbitrary impedance of the arbitrary standard type for standard 3 for channel 1.

`SENS2:CORR:COLL:CKIT:STAN2:ARB?` may return 100.3, which is the (Ω) value set for the arbitrary impedance of the arbitrary standard type for standard 2 for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C0

Set or query the $C_0$ value of the open standard type, which is set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN` for the specified calibration kit standard for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C0 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C0?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit. Range: 1 to 21.

If the specified value is outside the allowable setup range, an error occurs.

`<value>::=<NRf>` is the $C_0$ value of the open standard type for the specified calibration kit standard. Range: -1e18 to 1e18 fF; Units: fF (femto farad).

If the specified variable is outside the allowable setup range, the minimum or maximum value is set.

**Returns**

`<NRf>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN5:C0 120` sets 120 fF as the $C_0$ value of the open standard type for standard 5 on channel 1.

`SENS2:CORR:COLL:CKIT:STAN3:C0?` may return 125, which is means 125 fF (femto farad) is the value of $C_0$ for the open standard type set for standard 3 on channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C1

Set or query the $C_1$ value of the open standard type, which is set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN` for the specified calibration kit standard for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C1 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C1?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit. Range: 1 to 21.

If the specified value is outside the allowable setup range, an error occurs.

`<value>::=<NRf>` is the $C_1$ value of the open standard for the specified calibration kit standard. Range: -1e18 to 1e18; Units: 1e-27 farad/Hz.

If the specified variable is outside the allowable setup range, the minimum or maximum value is set.

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN3:C1 90` sets 90e-27 farad/Hz as the $C_1$ value of the open standard type for standard 3 to for channel 1.

`SENS3:CORR:COLL:CKIT:STAN2:C1?` may return 90, which means 90e-27 farad/Hz is the $C_1$ value of the open standard type for standard 2 on channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C2

Set or query the $C_2$ value of the open standard type, which is set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN` for the specified calibration kit standard for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C2 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C2?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

If the specified value is outside the allowable setup range, an error occurs.

`<value>::=<NRf>` is the $C_2$ value of the open standard for the specified calibration kit standard. Range: -1e18 to 1e18; Units: 1e-36 farad/Hz$^2$.

If the specified variable is outside the allowable setup range, the minimum value or the maximum value is set.

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN3:C2 12` sets 12e-36 farad/Hz$^2$ as the $C_2$ value of the open standard type for standard 3 on channel 1.

`SENS2:CORR:COLL:CKIT:STAN3:C2?` may return 15e-36, which means 15e36 farad/Hz$^2$ is the $C_2$ value of the open standard type for standard 3 on channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C3

Set or query the $C_3$ value of the open standard type, which is set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE OPEN` for the specified calibration kit standard for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C3 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:C3?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

If the specified value is outside the allowable setup range, an error occurs.

`<value>::=<NRf>` is the $C_3$ value of the specified calibration kit standard. Range: -1e18 to 1e18 (1e-45 farad/Hz$^3$).

If the specified variable is outside the allowable setup range, the minimum value or the maximum value is set.

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN6:C3 18` sets 18e-45 farad/Hz$^3$ as the $C_3$ value of the open standard type for standard 6 for channel 1.

`SENS3:CORR:COLL:CKIT:STAN2:C3?` may return 12e-45, which means 12e-045 farad/Hz$^3$ is the $C_3$ value of the open standard type for standard 2 for channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:CHARacter

Set or query the type of media for the specified calibration kit standard on the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:CHARacter { COAX
| WAV }
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:CHARacter?
```

**Related Commands** SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`COAXial` sets the coaxial media.

`WAVeguide` sets the waveguide media.

**Returns** `COAXial` means the type of media is set to coaxial.

`WAVeguide` means the type of media is set to waveguide.

**Examples** `SENS1:CORR:COLL:CKIT:STAN3:CHAR COAX` sets the coaxial media type for standard 3 on channel 1.

`SENS2:CORR:COLL:CKIT:STAN2:CHAR?` may return WAV, which means waveguide is the media type set for standard 2 on channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:DELay

Set or query the offset delay for the specified calibration kit standard for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:DELay <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:DELay?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the value of the offset delay (variable depending on the specified calibration kit standard). Range: -1e18 to 1e18.

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN3:DEL 13` sets the offset delay for standard 3 to 13 seconds for channel 1.

`SENS2:CORR:COLL:CKIT:STAN2:DEL?` may return 32, which is the offset delay (seconds) for standard 2 is set to 32 seconds for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FILE

Set or query the name of the touchstone file for the specified calibration kit standard for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FILE <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FILE?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<string>` is the name and destination of the touchstone file for the specified calibration standard.

**Returns** `<string>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN1:FILE "stan1.s2p"` sets the file *stan1.s2p* as the touchstone file for standard 1 on channel 1.

`SENS2:CORR:COLL:CKIT:STAN3:FILE?` may return *stan3.s2p*, which is the touchstone file set for standard 3 on channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMAXimum

Set or query the stopping frequency for the specified calibration kit standard for the specified channel. When you select waveguide as the media type, use this command to set the stopping frequency of the cutoff frequency.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMAXimum <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMAXimum?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the stop frequency for the specified calibration standard. Range: 0 to 1e12 (Hz).

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN1:FMAX 2e6` sets to 2 MHz for the stopping frequency value for standard 1 for channel 1.

`SENS3:CORR:COLL:CKIT:STAN2:FMAX?` may return 1e6, which means 1 MHz is the stopping frequency value for standard 2 on channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMINimum

Set or query the starting frequency for the specified calibration kit standard for the specified channel. When you select waveguide as the media type, use this command to set the starting frequency of the cutoff frequency.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMINimum <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:FMINimum?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the starting frequency for the specified calibration standard. Range: 0 to 1e12 (Hz).

**Returns** `<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN1:FMIN 1e6` sets the starting frequency for standard 1 to 1 MHz for channel 1.

`SENS4:CORR:COLL:CKIT:STAN2:FMIN?` may return 3e6, which means 3 MHz is the starting frequency value for standard 2 on channel 4.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L0

Set or query the value of $L_0$ for the short standard type set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt` for the specified calibration kit standard on the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L0 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L0?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the value of $L_0$ for the specified calibration standard. Range: -1e18 to 1e18; Units: picohenry (pH).

**Returns**

`<NRf>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:L0 1e3` sets the value of $L_0$ for the short standard type of standard 1 to 1e3 pH for channel 1.

`SENS4:CORR:COLL:CKIT:STAN3:L0?` may return 2e5, which means 2e5 pH is the value of $L_0$ for standard 3 of the short standard type on channel 4.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L1

Set or query the value of $L_1$ for the short standard type set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt` for the specified calibration kit standard on the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L1 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L1?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the value of $L_1$ for the specified calibration standard. Range: -1e18 to 1e18; Units: 1e-24 H/Hz.

**Returns**

`<NRf>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:L1 1e4` sets 1e4 (1e-24 H/Hz) as the value of $L_1$ for the short standard type for standard 1 on channel 1.

`SENS4:CORR:COLL:CKIT:STAN2:L1?` may return 2e4, which means 2e4 (1e-24 H/Hz) is the value of $L_1$ for standard 2 of the short standard type for channel 4.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L2

Set or query the value of $L_2$ for the short standard type set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt` for the specified calibration kit standard on the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L2 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L2?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the value of $L_2$ for the specified calibration standard. Range: -1e18 to 1e18; Units: 1e-33 H/Hz$^2$.

**Returns**

`<NRf>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:L2 2e5` sets 2e5 (1e-33 H/Hz$^2$) as the value of $L_2$ for the short standard type for standard 1 on channel 1.

`SENS2:CORR:COLL:CKIT:STAN3:L2?` may return 4e4, which means 4e4 (1e-33 H/Hz$^2$) is the value of $L_2$ for standard 2 of the short standard type for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L3

Set or query the value of $L_3$ for the short standard type set by `SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE SHORt` for the specified calibration kit standard on the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L3 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:L3?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the value of $L_3$ for the specified calibration standard. Range: -1e18 to 1e18; Units: 1e-42 H/Hz$^3$.

**Returns**

`<NRf>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:L3 1e6` sets 1e6 (1e-42 H/Hz$^3$) as the value of $L_3$ for standard 1 of the short standard type for channel 1.

`SENS3:CORR:COLL:CKIT:STAN2:L3?` may return 2e6, which means 2e6 (1e-42 H/Hz$^3$ ) is the value of $L_3$ for the short standard type for standard 2 for channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LABel

Set or query the label of the specified calibration kit standard for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LABel <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LABel?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<string>` is the label of the calibration kit standard.

Range: < 254 characters

**Returns**

`<string>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:LAB "SHORT3.5"` sets the label SHORT3.5 for standard 1 for channel 1.

`SENS2:CORR:COLL:CKIT:STAN2:LAB?` may return SHORT1.2, which is the label used for standard 2 for channel 2.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LOSS

Set or query the offset loss for the specified calibration kit standard for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LOSS <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:LOSS?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

`<value>::=<NRf>` is the offset loss of the calibration kit standard. Range: -1e18 to 1e18; Units: Ω/s.

**Returns**

`<NRf>`

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:LOSS 3000` sets 3 kΩ/s as the offset loss value of standard 1 for channel 1.

`SENS4:CORR:COLL:CKIT:STAN2:LOSS?` may return 4000, which means 4 kΩ/s is the offset loss value of standard 2 for channel 4.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE

Set or query standard type for the specified calibration kit standard for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE { OPEN | SHORt
| LOAD | THRU | UTHRu | ARBI | NONE }
```

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:TYPE?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the standard number of the calibration kit (1 to 21).

Choose one of the following standard types:

| Argument | Description |
|---|---|
| OPEN | Specifies the open standard. |
| SHORt | Specifies the short standard. |
| LOAD | Specifies the load standard. |
| THRU | Specifies the thru standard. |
| UTHRu | Specifies the unknown thru standard. |
| NONE | Specifies the ideal standard. |

**Returns**

See Arguments.

**Examples**

`SENS1:CORR:COLL:CKIT:STAN1:TYPE SHOR` specifies short as the standard type set for standard 1 on channel 1.

`SENS3:CORR:COLL:CKIT:STAN2:TYPE?` may return LOAD, which means load is the standard type set for standard 2 on channel 3.

## SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:Z0

Set or query the characteristic impedance for the specified calibration kit standard for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:Z0 <value>
SENSe<x>:CORRection:COLLect:CKIT:STAN<y>:Z0?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the number of the calibration kit standard (1 to 21).

`<value>::=<NRf>` is the characteristic impedance value of the calibration standard. Range: -1e18 to 1e18; Units: kΩ/s.

**Returns**

`<NRf>`

**Examples** `SENS1:CORR:COLL:CKIT:STAN1:Z0 40000` sets 40 kΩ/s as the characteristic impedance value of standard 1 for channel 1.

`SENS4:CORR:COLL:CKIT:STAN2:Z0?` may return 30000, which means 30 kΩ/s is the characteristic impedance value of standard 2 for channel 4.

## SENSe<x>:CORRection:COLLect:CLEar (No Query Form)

Clear the calibration measurement data for the specified channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:CLEar`

**Arguments** None

**Examples** `SENS1:CORR:COLL:CLE` clears the calibration data for channel 1.

## SENSe<x>:CORRection:COLLect:METHod:ERESPonse (No Query Form)

Set enhanced response calibration as the calibration method for the specified channel between the stimulus and response ports.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:METHod:ERESPonse <port1>,<port2>`

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN

SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt

SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD

SENSe<x>:CORRection:COLLect:[ACQuire]:THRU

SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<port1>::=<NR1>` is the response port (1 or 2).

`<port2>::=<NR1>` is the stimulus port (1 or 2).

**Examples**

In the following example, command 1 sets the calibration method to enhanced response, assigning port 1 as the response port and port 2 as the stimulus port. Commands 2, 3, 4, and 5 perform the calibration data measurements for the enhanced response calibration. Command 6 saves the output and applies the calibration data to the specified channel:

1. `SENS1:CORR:COLL:METH:ERES 2,1`
2. `SENS1:CORR:COLL:ACQ:OPEN 1`
3. `SENS1:CORR:COLL:ACQ:SHOR 1`
4. `SENS1:CORR:COLL:ACQ:LOAD 1`
5. `SENS1:CORR:COLL:ACQ:THRU 2,1`
6. `SENS1:CORR:COLL:SAVE`

## SENSe<x>:CORRection:COLLect:METHod:RESPonse:OPEN (No Query Form)

Set the response calibration type to open for the specified port and channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:METHod:RESPonse:OPEN <value>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN

SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port (1 or 2) where you perform the response calibration open.

**Examples**

In the following example, command 1 sets the response calibration to open for port 2. Commands 2 and 3 perform the calibration data measurements for the response calibration open. Command 4 saves the output and applies the calibration data to the specified channel. Command 5 queries the type of calibration used for the specified channel:

1. `SENS1:CORR:COLL:METH:RESP:OPEN 2`
2. `SENS1:CORR:COLL:ACQ:OPEN 2`
3. `SENS1:CORR:COLL:ACQ:LOAD 2`
4. `SENS1:CORR:COLL:SAVE`
5. `SENS1:CORR:COLL:METH:TYPE?`

## SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt

Set or query the response calibration type to short for the specified port and channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt <value>
SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt?
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt

SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port where you perform the response calibration (short).

**Returns**

1 or 2

**Examples** In the following example, command 1 sets the response calibration type to short for port 2. Commands 2 and 3 perform the calibration data measurements for the response calibration short. Command 4 saves the output and applies the calibration data to the specified channel. Command 5 queries the type of calibration used for the specified channel:

1. `SENS1:CORR:COLL:METH:RESP:SHOR 2`
2. `SENS1:CORR:COLL:ACQ:SHOR 2`
3. `SENS1:CORR:COLL:ACQ:LOAD 2`
4. `SENS1:CORR:COLL:SAVE`
5. `SENS1:CORR:COLL:METH:TYPE?`

## SENSe<x>:CORRection:COLLect:METHod:RESPonse:THRU (No Query Form)

Set the response calibration type to thru for the specified ports (response and stimulus) and the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:METHod:RESPonse:THRU
<port1>,<port2>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:[ACQuire]:THRU

SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<port1>::=<NR1>` is the response port (1 or 2).

`<port2>::=<NR1>` is the stimulus port (1 or 2).

**Examples** In the following example, command 1 sets the response calibration type to thru, assigning port 1 as the response port and port 2 as the stimulus port (1,2). Commands 2 and 3 perform the calibration data measurements for the response calibration thru. Command 4 saves the output and applies the calibration data to the specified channel. Command 5 queries the type of calibration used for the specified channel:

1. `SENS1:CORR:COLL:METH:RESP:THRU 1,2`
2. `SENS1:CORR:COLL:ACQ:THRU 1,2`
3. `SENS1:CORR:COLL:ACQ:ISOL 1,2`
4. `SENS1:CORR:COLL:SAVE`
5. `SENS1:CORR:COLL:METH:TYPE?`

## SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS (No Query Form)

Set the number of ports for the SOLT calibration for the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS <value>`

**Related Commands** SENSe<x>:CORRection:COLLect:METHod:SOLT:PORTS

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of ports set for the SOLT calibration.

**Examples** `SENS1:CORR:COLL:METH:SOLT:NPORTS 1` sets a 1-port SOLT calibration for channel 1.

`SENS2:CORR:COLL:METH:SOLT:NPORTS 2` sets a 2-port SOLT calibration for channel 2.

## SENSe<x>:CORRection:COLLect:METHod:SOLT:PORTS (No Query Form)

Set the ports for the SOLT calibration based on the number of ports set by `SENSe<x>:CORRection:COLLect:METHod:SOLT:NPORTS` for the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:METHod:SOLT:PORTS <array>`

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

**Needed for 1-2 ports:**

SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN

SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt

SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD

**Needed for 2 ports:**

SENSe<x>:CORRection:COLLect:[ACQuire]:THRU

**Optional for 2 ports:**

SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<array>` is the array of ports used for the SOLT calibration, where the number of ports is set by `SENS<x>:CORR:COLL:METH:SOLT:NPORT`.

**Examples**

**1-Port SOLT Calibration.** In the following example, command 1 sets a 1-port SOLT calibration. Command 2, sets the 1-port SOLT calibration to port 2. Commands 3, 4, and 5 perform the calibration data measurement for the 1-port SOLT calibration for port 2. Command 6 saves the output and applies the measurement data to the specified channel:

1. `SENS1:CORR:COLL:METH:SOLT:NPORTS 1`
2. `SENS1:CORR:COLL:METH:SOLT:PORTS 2`
3. `SENS1:CORR:COLL:ACQ:OPEN 2`
4. `SENS1:CORR:COLL:ACQ:SHOR 2`
5. `SENS1:CORR:COLL:ACQ:LOAD 2`
6. `SENS1:CORR:COLL:SAVE`

**2-Port SOLT Calibration.** In the following example, command 1 sets the number of ports (2) for the SOLT calibration. Command 2 assigns ports 1 as the response port and port 2 as the stimulus port (1,2) for the 2-port SOLT calibration. Commands 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12 perform the calibration data measurements for the 2-port SOLT calibration. Command 13 saves the output and applies the calibration data to the specified channel:

1. `SENS1:CORR:COLL:METH:SOLT:NPORTS 2`
2. `SENS1:CORR:COLL:METH:SOLT:PORTS 1,2`
3. `SENS1:CORR:COLL:ACQ:OPEN 1`
4. `SENS1:CORR:COLL:ACQ:OPEN 2`
5. `SENS1:CORR:COLL:ACQ:SHOR 1`
6. `SENS1:CORR:COLL:ACQ:SHOR 2`
7. `SENS1:CORR:COLL:ACQ:LOAD 1`
8. `SENS1:CORR:COLL:ACQ:LOAD 2`
9. `SENS1:CORR:COLL:ACQ:THRU 2,1`
10. `SENS1:CORR:COLL:ACQ:THRU 1,2`
11. `SENS1:CORR:COLL:ACQ:ISOL 2,1`
12. `SENS1:CORR:COLL:ACQ:ISOL 1,2`
13. `SENS1:CORR:COLL:SAVE`

## SENSe<x>:CORRection:COLLect:METHod:SOLT1 (No Query Form)

Set the calibration type to 1-port calibration for the specified port and channel.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:METHod:SOLT1 <value>`

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN

SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt

SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the port (1 or 2) at which you perform 1-port calibration.

**Examples**

In the following example, command 1 sets a 1-port SOLT calibration to port 2. Commands 2, 3, and 4 perform the calibration data measurements for the 1-port calibration. Command 5 saves the output and applies the calibration data to the specified channel:

1. `SENS1:CORR:COLL:METH:SOLT1 2` sets 1-port SOLT calibration for port 2 for channel 1.
2. `SENS1:CORR:COLL:ACQ:OPEN 2` sets the calibration type to response calibration (open) for port 2
3. `SENS1:CORR:COLL:ACQ:SHOR 2` sets the calibration type to response calibration (short) for port 2.
4. `SENS1:CORR:COLL:ACQ:LOAD 2` measures the calibration data for the (load) standard on port 2.
5. `SENS1:CORR:COLL:SAVE` calculates and applies the calibration coefficients according to the type of calibration and data measured

## SENSe<x>:CORRection:COLLect:METHod:SOLT2 (No Query Form)

Set the calibration type to full 2-port calibration between the specified ports for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:COLLect:METHod:SOLT2 <port1>,<port2>
```

**Related Commands**

SENSe<x>:CORRection:COLLect:CLEar

SENSe<x>:CORRection:COLLect:SAVE

SENSe<x>:CORRection:COLLect:METHod:TYPE?

SENSe<x>:CORRection:COLLect:[ACQuire]:OPEN

SENSe<x>:CORRection:COLLect:[ACQuire]:SHORt

SENSe<x>:CORRection:COLLect:[ACQuire]:LOAD

SENSe<x>:CORRection:COLLect:[ACQuire]:THRU

SENSe<x>:CORRection:COLLect:[ACQuire]:ISOLation

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<port1>::=<NR1>` is the first port (1 or 2).

`<port2>::=<NR1>` is the second port (1 or 2).

**Examples** In the following example, command 1 sets a 2-port SOLT calibration for ports 1 and 2, assigning port 1 as the response port and port 2 as the stimulus port. Commands 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11 perform the calibration measurements for the 2-port SOLT calibration. Command 12 saves the output and applies the calibration measurements to the specified channel:

1. `SENS1:CORR:COLL:METH:SOLT2 1,2`
2. `SENS1:CORR:COLL:ACQ:OPEN 1`
3. `SENS1:CORR:COLL:ACQ:OPEN 2`
4. `SENS1:CORR:COLL:ACQ:SHOR 1`
5. `SENS1:CORR:COLL:ACQ:SHOR 2`
6. `SENS1:CORR:COLL:ACQ:LOAD 1`
7. `SENS1:CORR:COLL:ACQ:LOAD 2`
8. `SENS1:CORR:COLL:ACQ:THRU 2,1`
9. `SENS1:CORR:COLL:ACQ:THRU 1,2`
10. `SENS1:CORR:COLL:ACQ:ISOL 2,1`
11. `SENS1:CORR:COLL:ACQ:ISOL 1,2`
12. `SENS1:CORR:COLL:SAVE`

## SENSe<x>:CORRection:COLLect:METHod:TYPE? (Query Only)

Query the type of calibration for the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:METHod:TYPE?`

**Returns** **Command variables.** <x> is the channel number (1 to 16).

| Response | Value |
|---|---|
| NONE | 0 |
| RESPO | 1 |
| RESPS | 2 |
| RESPT | 3 |
| ERES | 4 |
| SOLT | 5 |

**Examples** `SENS1:CORR:COLL:METH:TYPE?` may return 5, which means the calibration type selected for channel 1 is set to be SOLT.

## SENSe<x>:CORRection:COLLect:PARTial:SAVE (No Query Form)

Use this partial overwrite function to recalculate calibration coefficients according to the type of calibration and data measured. This action clears all measured data and the selection for calibration type.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:PARTial:SAVE`

**Arguments** None

**Examples** `SENS1:CORR:COLL:PART:SAVE` uses the partial overwrite function to recalculate calibration coefficients for channel 1.

## SENSe<x>:CORRection:COLLect:SAVE (No Query Form)

Calculate and apply the calibration coefficients based on the type of calibration and data measured. This action clears all measured data and the selection for calibration type.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Sense commands

**Syntax** `SENSe<x>:CORRection:COLLect:SAVE`

**Related Commands** SENSe<x>:CORRection:COLLect:METHod:TYPE?

SENSe<x>:CORRection:COLLect:METHod:ERESPonse

SENSe<x>:CORRection:COLLect:METHod:RESPonse:OPEN

SENSe<x>:CORRection:COLLect:METHod:RESPonse:SHORt

SENSe<x>:CORRection:COLLect:METHod:RESPonse:THRU

SENSe<x>:CORRection:COLLect:METHod:SOLT:PORTS

SENSe<x>:CORRection:COLLect:METHod:SOLT1

SENSe<x>:CORRection:COLLect:METHod:SOLT2

**Arguments** None

**Examples** `SENS1:CORR:COLL:SAVE` calculates and applies the calibration coefficients to channel 1.

## SENSe<x>:CORRection:EXTension:PORT<y>:FREQuency<z>

Set or query the frequency used to calculate the specified loss value for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:EXTension:PORT<y>:FREQuency<z> <value>
SENSe<x>:CORRection:EXTension:PORT<y>:FREQuency<z>?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2); <z> is the loss value (1 for Loss 1 or 2 for Loss 2).

`<value>::=<NRf>` is the frequency value used to calculate the specified loss value.

**Returns** `<NRf>`

**Examples** `SENS1:CORR:EXT:PORT1:FREQ1 5e6` sets frequency Loss 1 to 5 MHz for port 1 on channel 1.

`SENS3:CORR:EXT:PORT2:FREQ?` may return 6e6, which means 6 MHz is the value of the frequency Loss 2 set for port 2 on channel 3.

## SENSe<x>:CORRection:EXTension:PORT<y>:INCLude:[STATe]

Set or query the status of the respective port extension loss correction and frequency values for the specified port and channel. For Loss 2 to be enabled, Loss 1 must also be enabled.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:EXTension:PORT<y>:INCLude:[STATe]
<value>
SENSe<x>:CORRection:EXTension:PORT<y>:INCLude:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<boolean>`

**Returns** 1 means loss correction and frequency values are enabled.

0 means loss correction and frequency values are disabled

**Examples** `SENS1:CORR:EXT:PORT1:INCL:STAT 1` enables loss correction and frequency values for port 1 on channel 1.

`SENS3:CORR:EXT:PORT2:INCL:STAT?` may return 0, which means loss correction and frequency values are disabled for port 2 on channel 3.

## SENSe<x>:CORRection:EXTension:PORT<y>:LDC

Set or query the DC loss value for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:EXTension:PORT<y>:LDC <value>
SENSe<x>:CORRection:EXTension:PORT<y>:LDC?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NRf>` is the DC loss value of the port 1 or 2.

Range: -90 to 90 dB

**Returns** `<NRf>`

**Examples** `SENS1:CORR:EXT:PORT1:LDC 50` sets 50 dB as the DC loss value for port 1 on channel 1.

`SENS2:CORR:EXT:PORT2:LDC?` may return -30, which means -30 dB is the value of the DC loss for port 2 on channel 2.

## SENSe<x>:CORRection:EXTension:PORT<y>:LOSS<z>

Set or query the value of Loss 1 or Loss 2 for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:EXTension:PORT<y>:LOSS<z> <value>
SENSe<x>:CORRection:EXTension:PORT<y>:LOSS<z>?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2); <z> is the Loss number (1 or 2).

`<value>::=<NRf>` is the Loss value.

Range: -90 to 90 dB

**Returns** `<NRf>`

**Examples** `SENS1:CORR:EXT:PORT1:LOSS1 40` sets Loss 1 to 40 dB for port 1 on channel 1.

`SENS3:CORR:EXT:PORT2:LOSS2?` may return 60, which means 60 dB is the value of Loss 2 for port 2 on channel 3.

## SENSe<x>:CORRection:EXTension:PORT<y>:[TIME]

Set or query the delay time for the port extension for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:EXTension:PORT<y>:[TIME] <value>
SENSe<x>:CORRection:EXTension:PORT<y>:[TIME]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NRf>` is the delay time for the port extension.

Range: -10 to 10 s

**Returns** `<NRf>`

**Examples** `SENS1:CORR:EXT:PORT1:TIME 4` sets 4 seconds as the delay time to the port extension of port 1 for channel 1.

`SENS4:CORR:EXT:PORT2:TIME?` may return -2, which means -2 s is the delay time to the port extension of port 2 for channel 4.

## SENSe<x>:CORRection:EXTension:[STATe]

Set or query the state of the port extension for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:EXTension:[STATe] <value>
SENSe<x>:CORRection:EXTension:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means port extension is enabled.

0 means port extension is disabled.

**Examples** `SENS1:CORR:EXT:STAT 1` enables port extension for channel 1.

`SENS2:CORR:EXT:STAT?` may return 0, which means port extension is disabled for channel 2.

## SENSe<x>:CORRection:RECeiver<y>:COLLect:[ACQuire] (No Query Form)

Execute the receiver calibration for the specified port and channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:RECeiver<y>:COLLect:[ACQuire] <value>
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NR1>` is the source port (1 or 2) for the receiver calibration.

**Examples**

`SENS1:CORR:REC1:COLL:ACQ 2` executes the receiver calibration for port 1 on channel 1, using port 2 as the source port.

`SENS3:CORR:REC2:COLL:ACQ 1` executes the receiver calibration for port 2 on channel 3, using port 1 as the source port.

## SENSe<x>:CORRection:RECeiver<y>:[STATe]

Set or query the state of error correction for the receiver calibration for the specified port and channel.

***NOTE.*** *You must perform the receiver calibration before you can enable or disable error correction for that port.*

**Group**

Sense commands

**Syntax**

```
SENSe<x>:CORRection:RECeiver<y>:[STATe] <value>
SENSe<x>:CORRection:RECeiver<y>:[STATe]?
```

**Related Commands**

SENSe<x>:CORRection:RECeiver<y>:COLLect:[ACQuire]

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<boolean>`

**Returns**

1 means error correction is enabled.

0 means error correction is disabled.

**Examples**

In the following example, command 1 performs the receiver calibration for port 1, by using port 2 as the source port. Command 2 enables error correction for the receiver calibration for the specified port and channel. Command 3 queries whether error correction is enabled or disabled for the receiver calibration for a specified port:

1. `SENS1:CORR:REC1:COLL:ACQ 2`
2. `SENS1:CORR:REC1:STAT 1`
3. `SENS1:CORR:REC1:STAT?`

## SENSe<x>:CORRection:RVELocity:COAX

Set or query the velocity factor for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:RVELocity:COAX <value>
SENSe<x>:CORRection:RVELocity:COAX?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the velocity factor value.

Range: 0 to 10

**Returns** `<NRf>`

**Examples** `SENS1:CORR:RVEL:COAX 0.3` sets 0.3 as the velocity factor value for channel 1.

`SENS2:CORR:RVEL:COAX?` may return 1, which means 1 is the velocity factor value set for channel 2.

## SENSe<x>:CORRection:[STATe]

Set or query the error correction status for the specified channel.

**Command variables.** <x> is the channel number (1 to 16).

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:[STATe] <value>
SENSe<x>:CORRection:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means error correction is enabled.

0 means error correction is disabled.

**Examples** `SENS1:CORR:STAT 1` enables error correction for channel 1.

`SENS2:CORR:STAT?` may return 0, which means error correction is disabled for channel 2.

## SENSe<x>:CORRection:TRIGger:FREE

Enable or disable the internal trigger source for calibrations for the specified channel.

***NOTE.*** *Changing the trigger source during a sweep may cancel the sweep.*

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:TRIGger:FREE <value>
SENSe<x>:CORRection:TRIGger:FREE?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns** 1 means the trigger source is set to internal.

0 means the trigger source is set to system.

**Examples** `SENS1:CORR:TRIG:FREE 1` sets the trigger source to internal for channel 1.

`SENS2:CORR:TRIG:FREE?` may return 0, which means the trigger source is set to system for channel 2.

## SENSe<x>:CORRection:TYPE? (Query Only)

Query the calibration type used for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:CORRection:TYPE?
```

**Related Commands** SENSe<x>:CORRection:COLLect:METHod:TYPE?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

| Value | Description | Calibration |
|---|---|---|
| 0 | Enhanced response calibration | ERES |
| 1 | No calibration | NONE |
| 2 | Response calibration (open) | RESPO |
| 3 | Response calibration (short) | RESPS |
| 4 | Response calibration (thru) | RESPT |
| 5 | SOLT calibration | SOLT |

**Examples** `SENS1:CORR:TYPE?` may return 1, if no calibration data exists for channel 1.

`SENS2:CORR:TYPE SOLT` sets SOLT calibration for channel 2.

## SENSe<x>:FREQuency:CENTer

Set or query the center frequency value of the sweep range for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:FREQuency:CENTer <value>
SENSe<x>:FREQuency:CENTer?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the center frequency value (Hz) of the sweep range.

Range: 0 to [maximum frequency 10e5] Hz

**Returns** `<NRf>`

**Examples** `SENS1:FREQ:CENT 1e6` sets the center frequency value to 1 MHz for channel 1.

`SENS2:FREQ:CENT?` may return 2e6, which means 2 MHz is the center frequency value set for channel 2.

## SENSe<x>:FREQuency:CW

Set or query the CW frequency value of the sweep range of the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:FREQuency:CW <value>
SENSe<x>:FREQuency:CW ?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the CW frequency value (Hz) of the sweep range.

**Returns** `<NRf>`

**Examples** `SENS1:FREQ:CW 3.525e6` sets the CW value for channel 1 to 3.525 MHz.

`SENS2:FREQ:CW?` may return 3.525e6, which means the CW value is set to 3.525 MHz for channel 2.

## SENSe<x>:FREQuency:DATA? (Query Only)

Query the frequency at all measurement points for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:FREQuency:DATA?
```

**Related Commands** SENSe<x>:SWEep:POINts

SENSe<x>:FREQuency:STARt

SENSe<x>:FREQuency:STOP

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`DATA` is an array of frequency values.

If there are NP measurement points, `DATA[n-1]` is the frequency at the $n^{th}$ measurement point.

Range of *n*: 1 to NP

**Examples** `SENS1:FREQ:DATA?` may return 1e9, 2e9, 3e9, if there are only 3 measurement points with a linear sweep range from 1 to 3 GHz for channel 1.

## SENSe<x>:FREQuency:SPAN

Set or query the span of the sweep range for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:FREQuency:SPAN <value>
SENSe<x>:FREQuency:SPAN?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the span of the sweep range.

Range: 0 [to maximum frequency -10e5] Hz.

**Returns** `<NRf>`

**Examples** `SENS1:FREQ:SPAN 2e6` sets 2 MHz as the span value of the sweep range for channel 1.

`SENS3:FREQ:SPAN?` may return 3e6, which means 3 MHz is the span value of the sweep range set for channel 3.

## SENSe<x>:FREQuency:STARt

Set or query the value of the starting frequency of the sweep range for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:FREQuency:STARt <value>
SENSe<x>:FREQuency:STARt?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the starting frequency of the sweep range.

Range: 1e5 to [maximum frequency] Hz

**Returns** `<NRf>`

**Examples**

`SENS1:FREQ:STAR 1e5` sets the starting frequency value of the sweep range to 100 kHz for channel 1.

`SENS2:FREQ:STAR?` may return 4e5, which means 400 kHz is the starting frequency value of the sweep range set for channel 2.

## SENSe<x>:FREQuency:STOP

Set or query the value of the stopping frequency of the sweep range for the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:FREQuency:STOP <value>
SENSe<x>:FREQuency:STOP?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the stopping frequency value of the sweep range.

Range: 1e5 to [maximum frequency] Hz

**Returns**

`<NRf>`

**Examples**

`SENS1:FREQ:STOP 3e8` sets 300 MHz as the stopping frequency value of the sweep range for channel 1.

`SENS2:FREQ:STOP?` may return 4e8, which means 400 MHz is the stopping frequency value of the sweep range set for channel 2.

## SENSe<x>:SEGMent:DATA

Create or query the array for the segment sweep table on the specified channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:SEGMent:DATA SEGTABLE
SENSe<x>:SEGMent:DATA?
```

**Related Commands**

MMEMory:STORe:SEGMent

## Arguments

**Command variables.** <x> is the channel number (1 to 16).

SEGTABLE is the array used to define the segment sweep tabled with the following parameters: <mode>,<ifbw>,<power>,<delay>,<sweep>,<time>,<row (n)>,<start n>,<stop n>,<nop n>,<ifbw n>,<power n>,<delay n>,<sweep n>,<time n>, where *n* represents the number of times this parameter appears.

Optional parameters <ifbw n>,<power n>, <delay n>,<sweep n>,<time n> only appear if enabled.

If an optional parameter is enabled, all segments must contain a valid value for this parameter. If an optional parameter is not enabled, the array will not include its respective parameter *n*. Segments cannot overlap in frequency range. If an invalid parameter is set, an error occurs.

| Parameter | Description | Value |
|---|---|---|
| mode | 0 or OFF: Center/Span | Boolean |
| Ifbw | 0 or OFF: Disable IF BW<br>1 or ON: Enable IF BW | Boolean |
| power | 0 or OFF: Disable Power Level<br>1 or ON: Disable Power Level | Boolean |
| delay | 0 or OFF: Disable Delay<br>1 or ON: Enable Delay | Boolean |
| sweep | 0 or OFF: Disable Sweep Mode<br>1 or ON: Enable Sweep Mode | Boolean |
| time | 0 or OFF: Disable time<br>1 or ON: Enable time | Boolean |
| row n | Number of segments n | <NR1> |
| Start n | Start/Center value for segment n | <NRf><br>Range: Min to Max Freq |
| Stop n | Stop/Span value for segment n | <NRf><br>Range: Min to Max Freq |
| nop n | Number of points n. Max. combined total for all segments must be less than 20,002 measurement points. | <NR1><br>Range: 1 to 20,001 |
| Ifbw n | Optional: IF bandwidth value (Hz) for segment n | <NR1><br>Range: 1 to 5e5 Hz |
| power n | Optional: Power level value (dBm) for segment n | <NRf><br>Range: –50 to 10 dBm |
| delay n | Optional: Sweep delay time (seconds) for segment n. Auto Sweep Time must be disabled. | <NRf><br>Range 0 to 1 s |

| Parameter | Description | Value |
|---|---|---|
| sweep n | Optional: Sweep mode for segment n | Normal mode: STEPped<br>Fast mode: FSTepped |
| time n | Optional: Sweep time (seconds) for segment n. Auto Sweep Time must be disabled. | <NRf><br>Range: 1e-3 to 2e3 |

**Returns** See Arguments.

**Examples** `SENS1:SEGM:DATA 1,0,0,0,0,1,1,1e9,2e9,101,1` sets a single segment with a sweep range starting at 1 GHz and stopping at 2 GHz, with 101 points for channel 1. This segment has a sweep time of 1 second if auto sweep time is disabled.

`SENS1:SEGM:DATA 0,1,1,1,1,1,2,1.5e9,1e9,101,1000,-10,0.5,STEP,0.1,5.5e9,1e9,123,2000,5,1,FST,3.5` sets 2 different segments using center and span to set the range for channel 1. All optional parameters IF BW, power level, delay, sweep mode, and time are enabled. As a result, each segment must set its respective optional parameter.

The first segment has a sweep range between 1 and 2 GHz, where the center frequency is 1.5 GHz, with a span of 1 GHz with 101 measurement points. IF BW is set to 1-kHz, power level to -10 dBm, sweep delay to 500 ms, sweep mode to normal with a minimum sweep time of 0.1 ms.

The second segment has a sweep range between 5 and 6 GHz (center frequency of 5.5 GHz with a span of 1 GHz) with 123 measurement points. IF BW is set to 2 kHz, power level to 5 dBm, sweep delay to 1 second, sweep mode to fast with a minimum sweep time of 3.5 seconds. Sweep delay and time is dependent on auto sweep time being disabled.

`SENS1:SEGM:DATA?` may return 0, 0, 0, 0, 0, 0, 1, 300000, 0, 2, which are the preset values for the segment table, with a center frequency of 300 kHz, 0 span, and 2 measurement points.

## SENSe<x>:SEGMent:SWEep:POINts? (Query Only)

Query the total number of measurement points for the segment sweep for the specified channel.

**Group** Sense commands

**Syntax** `SENSe<x>:SEGMent:SWEep:POINts?`

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NRf>`

Range: 1 to 20,001. Preset default value is 2.

**Examples** `SENS1:SEGM:SWE:POIN?` may return 2, which is the total number of measurement points for the segment sweep for channel 1.

## SENSe<x>:SEGMent:SWEep:TIME:[DATA]? (Query Only)

Query the total sweep time (excluding sweep delay) of the segmented sweep for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:SEGMent:SWEep:TIME:[DATA]?
```

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NRf>`

**Examples** `SENS1:SEGM:SWE:TIME:DATA?` may return 0.178071, which means the total sweep time is set to 0.178071 seconds (preset value), excluding sweep delay for the segment sweep for channel 1.

## SENSe<x>:SWEep:DELay

Set or query the sweep delay for the specified channel.

***NOTE.*** *You must disable auto sweep time before using this command.*

**Group** Sense commands

**Syntax**

```
SENSe<x>:SWEep:DELay <value>
SENSe<x>:SWEep:DELay?
```

**Related Commands** SENSe<x>:SWEep:TIME:AUTO

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the sweep delay in seconds.

Range: 0 to 1

**Returns** `<NRf>`

**Examples** `SENS1:SWE:DEL 0.5` sets the sweep delay to 0.5 seconds for channel 1.

`SENS2:SWE:DEL?` may return 0.2, which means the sweep delay is set to 0.2 seconds for channel 2.

## SENSe<x>:SWEep:GENeration

Set or query the sweep mode of the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:SWEep:GENeration { STEPped | FSTepped }
SENSe<x>:SWEep:GENeration?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`STEPped` is the normal sweep mode.

`FSTepped` is the fast sweep mode.

**Returns** `STEPped` means the normal sweep mode is set.

`FSTepped` means the fast sweep mode is set.

**Examples** `SENS1:SWE:GEN FST` sets the sweep mode to fast for channel 1.

`SENS2:SWE:GEN?` may return STEPped, which means the sweep mode is set to normal for channel 2.

## SENSe<x>:SWEep:POINts

Set or query the number of measurement points for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:SWEep:POINts <value>
SENSe<x>:SWEep:POINts?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of measurement points.

Range: 0 to 20,001

**Returns**

`<NR1>`

**Examples**

`SENS1:SWE:POIN 201` sets 201 measurement points for channel 1.

`SENS2:SWE:POIN?` may return 450, which is the number of measurement points set for channel 2.

## SENSe<x>:SWEep:TIME:AUTO

Set or query the status of the automatic sweep time for the selected channel.

**Group**

Sense commands

**Syntax**

```
SENSe<x>:SWEep:TIME:AUTO <value>
SENSe<x>:SWEep:TIME:AUTO?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns**

1 means automatic sweep time is enabled.

0 means automatic sweep time is disabled.

**Examples**

`SENS1:SWE:TIME:AUTO 1` enables automatic sweep time for channel 1.

`SENS3:SWE:TIME:AUTO?` may return 0, which means automatic sweep time is disabled for channel 3.

## SENSe<x>:SWEep:TIME:DATA

Set or query the value of the sweep time for the specified channel. If the specified time value (in seconds) is outside the range, the respective minimum or maximum value is set accordingly.

The range varies based on measurement conditions such as IF BW and the number of measurement points.

***NOTE.*** *You must disable auto sweep time before using this command.*

**Group** Sense commands

**Syntax**

```
SENSe<x>:SWEep:TIME:DATA <value>
SENSe<x>:SWEep:TIME:DATA?
```

**Related Commands** SENSe<x>:SWEep:TIME:AUTO

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the sweep time (in seconds) for the specified channel.

Range: 1 to 20,001

Preset default value is 2.

Range: Varies based on measurement conditions. Absolute min/max range is = 1e-3 to 2e3.

**Returns** `<NRf>`

**Examples** `SENS1:SWE:TIME:DATA 0.01` may set the sweep time value for channel 1 to 0.01 seconds, if 0.01 seconds is within the acceptable time range.

`SENS2:SWE:TIME:DATA?` may return 0.02, which means 0.02 seconds is the sweep time value set for channel 2, if 0.02 seconds is within the acceptable time range.

## SENSe<x>:SWEep:TYPE

Set or query the sweep type for the selected channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>:SWEep:TYPE { LINear | LOGarithmic | SEGMent | POWer
}
SENSe<x>:SWEep:TYPE?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

| Argument | Description |
|---|---|
| LINear | Indicates the linear sweep. |
| LOGarithmic | Indicates the logarithmic sweep. |
| SEGMent | Indicates the segmented sweep. |
| POWer | Indicates the power sweep. |

**Returns** See Arguments.

**Examples** `SENS1:SWE:TYPE LIN` sets the sweep type for channel 1 to linear sweep.

`SENS2:SWE:TYPE?` may return POWer, which is the sweep type used for channel 2.

## SENSe<x>FREQuency:FIXed

Set or query the fixed frequency value used by power sweep for the specified channel.

**Group** Sense commands

**Syntax**

```
SENSe<x>FREQuency:FIXed <value>
SENSe<x>FREQuency:FIXed?
```

**Arguments** `<value>::=<NRf>` is the fixed value (Hz) used by power sweep.

**Returns** `<NRf>`

**Examples** `SENS1:FREQ:FIX 1e9` sets 1 GHz as the fixed value used by power sweep for channel 1.

`SENS3:FREQ:FIX?` may return 3e9, which means 3 GHz is the fixed value used by power sweep for channel 3.

## SERVice:CHANnel:ACTive

Set or query the active channel.

***NOTE.*** *An error occurs if the specified channel is not yet allocated.*

**Group** Service commands

**Syntax**

```
SERVice:CHANnel:ACTive <value>
SERVice:CHANnel:ACTive?
```

**Related Commands** SERVice:CHANnel<x>:COUNt

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of the active channel.

**Returns** `<NR1>`

**Examples** `SERV:CHAN:ACT 2` sets channel 2 as the active channel.

`SERV:CHAN:ACT?` may return 3, which means channel 3 is the active channel.

## SERVice:CHANnel:TRACe:COUNt? (Query Only)

Query the number of traces per channel.

**Group** Service commands

**Syntax**

```
SERVice:CHANnel:TRACe:COUNt?
```

**Returns** `<NR1>`

**Examples** `SERV:CHAN:TRAC:COUN?` may return 4, which means there are 4 traces per channel.

## SERVice:CHANnel<x>:COUNt

Set or query the number of allocated channels.

**Group** Service commands

**Syntax**

```
SERVice:CHANnel<x>:COUNt <value>
SERVice:CHANnel<x>:COUNt?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the number of allocated channels.

**Returns** `<NR1>`

**Examples** `SERV:CHAN:COUN?` may return 2, which means there are 2 allocated channels.

`SERV:CHAN3:COUN` allocates channel 3.

## SERVice:CHANnel<x>:TRACe:ACTive? (Query Only)

Query the active trace for the specified channel.

**Group** Service commands

**Syntax** `SERVice:CHANnel<x>:TRACe:ACTive?`

**Related Commands** SERVice:CHANnel<x>:COUNt

CALCulate<x>:PARameter:COUNt

CALCulate<x>:PARameter<y>:[SELect]

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** `SERV:CHAN1:TRAC:ACT?` may return 4, which means trace 4 is the active trace on channel 1.

## SIMulator:FILE:FET (No Query Form)

Load the CSV file of the factory error terms to the specified port.

**Group** Simulator commands

**Syntax**

```
SIMulator:FILE:FET <value1>,<value2>
SIMulator:FILE:FET?
```

**Arguments** `<value1>::=<NR1>` is the port number (1 or 2).

`<value2>::=<string>` is the CSV file containing factory error terms.

**Returns** `<NR1>,<string>`

**Examples** `SIM:FILE:FET "1,FactoryErrorTermsSynthPort1.csv"` loads factory error terms from the file *FactoryErrorTermsSynthPort1.csv* to port 1.

## SIMulator:FILE:FOLDer

Set or query the name of the folder to be searched for simulator files. This command does not create a new folder.

**Group** Simulator commands

**Syntax**

```
SIMulator:FILE:FOLDer <value>
SIMulator:FILE:FOLDer?
```

**Arguments** `<value>::=<string>` is the simulator files folder.

**Returns** `<string>`

**Examples** `SIM:FILE:FOLD "C:\ProgramData\Tektronix\VectorVuPC\SimulatorDataFiles"` assigns *SimulatorDataFiles* as the folder to be searched for simulator files.

`SIM:FILE:FOLD?` may return *SimulatorDataFiles*, which is the simulator files folder.

## SIMulator:FILE:LCOM (No Query Form)

Load the simulator loss compensation file.

**Group** Simulator commands

**Syntax** `SIMulator:FILE:LCOM <value>`

**Arguments** `<value>::=<string>` is the simulator loss compensation file.

**Examples** `SIM:FILE:LCOM "SimulatorLossCompensation.s2p"` loads the simulator loss compensation file *SimulatorLossCompensation.s2p*.

## SIMulator:FILE:PCF (No Query Form)

Load the simulator PCF file.

**Group** Simulator commands

**Syntax** `SIMulator:FILE:PCF <value>`

**Arguments** `<value>::=<string>` is the simulator PCF file.

**Examples** `SIM:FILE:PCF "Simulator2.pcf"` loads the file *Simulator2.pcf*.

## SIMulator:FILE:RTF (No Query Form)

Load the simulator RTF file.

**Group** Simulator commands

**Syntax** `SIMulator:FILE:RTF <value>`

**Arguments** `<value>::=<string>` is the simulator RTF file.

**Examples** `SIM:FILE:RTF "Simulator1.rtf"` loads the *Simulator1.rtf*.

## SIMulator:FILE:STF (No Query Form)

Load the simulator STF file.

**Group** Simulator commands

**Syntax** `SIMulator:FILE:STF <value>`

**Arguments** `<value>::=<string>` is the simulator STF file.

**Examples** `SIM:FILE:STF "Simulator1.stf"` loads the simulator STF file *Simulator1.stf*.

## SIMulator:FILE:UET (No Query Form)

Load the CSV file containing user error terms to the specified port.

**Group** Simulator commands

**Syntax**

```
SIMulator:FILE:UET <value1>,<value2>
```

**Arguments** `<value1>::=<NR1>` is the specified port (1 or 2).

`<value2>::=<string>` is the CSV file containing user error terms.

**Examples** `SIM:FILE:UET "2,UserErrorTermsPort2.csv"` loads user error terms from the file *UserErrorTermsPort2.csv* to port 2.

## SIMulator:FILEname

Set or query the SNP (touchstone) file name that describes the device under test (DUT).

**Group** Simulator commands

**Syntax**

```
SIMulator:FILEname <value>
SIMulator:FILEname?
```

**Arguments** `<value>::=<string>` is the SNP (touchstone) file name describing the DUT.

**Returns** `<string>`

**Examples** `SIM:FILE "SimulatorFilter.s2p"` loads the file *SimulatorFilter.s2p* , which describes the DUT.

## SIMulator:NF

Set or query the noise floor value for simulator mode.

**Group** Simulator commands

**Syntax**

```
SIMulator:NF <value>
SIMulator:NF?
```

**Arguments** `<value>::=<NRf>` is the noise floor value for simulator mode.

Range: -150 to 0 dBm/Hz

**Returns** `<NRf>`

**Examples** `SIM:NF -100` sets the noise floor value to -100 dBm/Hz for simulator mode.

`SIM:NF?` may return -99, which means -99 dBm/HZ is the noise floor value for simulator mode.

## SIMulator:RCF

Set or query the reference calibration factor of the instrument.

**Group** Simulator commands

**Syntax**

```
SIMulator:RCF <value>
SIMulator:RCF?
```

**Arguments** `<value>::=<NRf>` is the reference calibration factor of the instrument.

Range: 1 to 150

**Examples** `SIM:RCF 100` sets the reference calibration factor of the instrument to 100.

`SIM:RCF?` may return 50, which is the reference calibration factor of the instrument.

## SOURce<x>:POWer:CENTer

Set or query the center value for the power sweep on the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:CENTer <value>
SOURce<x>:POWer:CENTer?
```

**Related Commands** SOURce<x>:POWer:SPAN

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the center value (in dBm) for power sweep for the specified channel.

**Returns** `<NRf>`

**Examples** `SOUR1:POW:CENT 0` sets the center value for the power sweep to 0 dBm on channel 1.

`SOUR5:POW:CENT?` may return 2, which means the center value of the power sweep is set to 2 dBm on channel 5.

## SOURce<x>:POWer:[LEVel]:[IMMediate]:[AMPLitude]

Set or query the value of the power level on the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:[LEVel]:[IMMediate]:[AMPLitude] <value>
SOURce<x>:POWer:[LEVel]:[IMMediate]:[AMPLitude]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the power level value (dBm) for the specified channel.

**Returns** `<NRf>`

**Examples**

`SOUR1:POW:LEV:IMM:AMPL 10` sets the power level value to 10 dBm for channel 1.

`SOUR9:POW:LEV:IMM:AMPL?` may return 5, which means the power level value is set to 5 dBm for channel 9.

## SOURce<x>:POWer:[LEVel]:SLOPe:[DATA]

Set or query the correction value for the power slope for the specified channel.

**Group**

Source commands

**Syntax**

```
SOURce<x>:POWer:[LEVel]:SLOPe:[DATA] <value>
SOURce<x>:POWer:[LEVel]:SLOPe:[DATA]?
```

**Related Commands**

SOURce<zx>:POWer:[LEVel]:SLOPe:[STATe]

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the correction value for the power slope.

Range: -2 to 2 dB/GHz

**Returns**

`<NRf>`

**Examples**

`SOUR1:POW:LEV:SLOP:DATA 0.2` sets the correction value of the power slope to 0.2 dB/GHz on channel 1.

`SOUR4:POW:LEV:SLOP:DATA?` may return -0.1, which means -0.1 dB/GHz is the correction value for the power slope on channel 4.

## SOURce<x>:POWer:PORT<y>:CORRection:COLLect:[ACQuire] (No Query Form)

Set the power calibration using data from power sensor **A** for the specified port and channel.

When you complete the calibration measurements, this command turns on the power level error correction.

***NOTE.*** *An error occurs if the power sensor is not properly connected.*

**Group**

Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:[ACQuire] ASENsor
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`ASENsor` is power sensor **A**, which sets power calibration using data from power sensor **A** for a specified port.

**Examples**

`SOUR1:POW:PORT1:CORR:COL:ACQ ASEN` sets power calibration using data from power sensor **A** (ASENsor) for port 2 on channel 1.

## SOURce<x>:POWer:PORT<y>:CORRection:COLLect:ASENsor:RCFactor

Set or query the value of the reference calibration coefficient for the active power sensor for the specified channel. The reference calibration coefficient is the calibration coefficient at 50 MHz.

**Group**

Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:ASENsor:RCFactor
<value>
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:ASENsor:RCFactor?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NRf>` is the RC factor for the sensor calibration. Range: 1 to 150%.

**Returns**

`<NRf>`

**Examples**

`SOUR1:POW:PORT1:CORR:COLL:ASEN:RCF 5` sets the reference calibration coefficient to 5% for port 1 on channel 1.

`SOUR2:POW:PORT2:CORR:COLL:ASEN:RCF?` may return 30, which means 30% is the reference calibration coefficient set for port 2 on channel 2.

## SOURce<x>:POWer:PORT<y>:CORRection:COLLect:NTOLerance

Set or query the tolerance value for power calibration data of the specified port and channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:NTOLerance
<value>
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:NTOLerance?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NR1>` is the tolerance value of power calibration data. Range: 1 to 100 dB.

**Returns** `<NR1>`

**Examples** `SOUR1:POW:PORT2:CORR:COLL:NTOL 3` sets the tolerance value of power calibration data to 3 dB for port 2 on channel 1.

`SOUR2:POW:PORT1:CORR:COLL:NTOL?` may return 3, which means 3 dB is the tolerance value of power calibration data for port 1 on channel 2.

## SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:ASENsor:DATA

Set or query the calibration table array for the active power sensor of the specified port at the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:ASENsor:
DATA <DATA>
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:ASENsor:
DATA?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

is the array that represents the calibration table with (1+2N) elements, where N is the number of set data items. An error occurs if the `DATA` array does not contain (1+2N) elements.

For any integer *n* between 1 and N:

`DATA[0]` is the number of data items to set (1 to 100).

`DATA[2n-1]` is the frequency of the $n^{th}$ data item (1 kHz to 5 GHz).

`DATA[2n]` is the calibration coefficient of the $n^{th}$ data item (1% to 150%).

**Returns** `<NRf>`

**Examples** `SOUR1:POW:PORT1:CORR:COLL:TABL:ASEN:DATA 4,1e6,96.5,1e7,97,1e8,99,2e8,99` sets the data array for the calibration table with the assigned values for port 1 on channel 1.

`SOUR3:POW:PORT2:CORR:COLL:TABL:ASEN:DATA?` may return 4, 1e6, 96.5, 1e7, 97, 1e8, 99, 2e8, 99, which is the data array that represents the calibration table for port 2 on channel 3.

## SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:DATA

Set or query the data array that represents the loss compensation table for the active power sensor at the specified port for the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:DATA
<DATA>
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:DATA?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`DATA` is an array that represents the loss compensation table with (1+2N) elements, where N is the number of set data items. An error occurs if the `VALUE` array does not contain (1+2N) elements.

For any integer *n* between 1 and N:

`DATA[0]` is the number of data items to set. Range: 1 to 100 dB.

`DATA[2n-1]` is the frequency of the $n^{th}$ data item (1 kHz to 5 GHz).

`DATA[2n]` is the loss of the $n^{th}$ data item.

**Returns** `<NRf>`

**Examples** `SOUR1:POW:PORT1:CORR:COLL:TABL:LOSS:DATA 0,5,6` sets the data array for the loss compensation table with the assigned values for port 1 on channel 1.

`SOUR2:POW:PORT2:CORR:COLL:TABL:LOSS:DATA?` may return 1, 2, 3, 4, 5, which is the data array for the loss compensation table with the assigned values for port 2 on channel 2.

## SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:[STATe]

Set or query the loss compensation for the specified port and channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:
[STATe] <value>
SOURce<x>:POWer:PORT<y>:CORRection:COLLect:TABLe:LOSS:
[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<boolean>`

**Returns** 1 means loss compensation is enabled.

0 means loss compensation is disabled.

**Examples** `SOUR1:POW:PORT1:CORR:COLL:TABL:LOSS:STAT 1` enables loss compensation for port 1 on channel 1.

`SOUR2:POW:PORT2:CORR:COLL:TABL:LOSS:STAT?` may return 0, which means loss compensation is disabled for port 2 on channel 2.

## SOURce<x>:POWer:PORT<y>:CORRection:DATA

Set or query the power calibration data array, for the specified port and channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:DATA <DATA>
SOURce<x>:POWer:PORT<y>:CORRection:DATA?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`DATA` is the power calibration data array with N number of points. For any value *n* between 1 and N.

`DATA[n-1]` is the data at the $n^{th}$ measurement point.

**Returns** `<NRf>`

**Examples** `SOUR1:POW:PORT1:CORR:DATA?` may return 4e9, which means 4 GHz is the value of the power calibration data array set for port 1 on channel 1.

`SOUR3:POW:PORT2:CORR:DATA 6e9` sets the power calibration data array to 6 GHz for port 2 on channel 3.

## SOURce<x>:POWer:PORT<y>:CORRection:[STATe]

Set or query the state of the power level error correction for the specified port and channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:CORRection:[STATe] <value>
SOURce<x>:POWer:PORT<y>:CORRection:[STATe]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<boolean>`

**Returns** 1 means power level error correction is turned on.

0 means power level error correction is turned off.

**Examples** `SOUR1:POW:PORT1:CORR:STAT 1` enables power level error correction for port 1 on channel 1.

`SOUR3:POW:PORT2:CORR:STAT?` may return 0, which means power level error correction is disabled for port 2 on channel 3.

## SOURce<x>:POWer:PORT<y>:COUPle

Set or query the same power level for all ports on the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:COUPle <value>
SOURce<x>:POWer:PORT<y>:COUPle?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<boolean`

**Returns** 1 means all ports are set to output the same power level.

0 means all ports are set to output individual power levels.

**Examples** `SOUR1:POW:PORT1:COUP 1` sets all ports on channel 1 to output the same power level.

`SOUR6:POW:PORT1:COUP?` may return 0, which means all ports on channel 6 are set to output individual power levels.

## SOURce<x>:POWer:PORT<y>:[LEVel]:[IMMediate]:[AMPLitude]

Set or query the value for the power level for the specified port and channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:PORT<y>:[LEVel]:[IMMediate]:[AMPLitude]
<value>
SOURce<x>:POWer:PORT<y>:[LEVel]:[IMMediate]:[AMPLitude]?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NRf>` is the power level (dBm) for the specified port and channel.

**Returns** `<NRf>`

**Examples** `SOUR1:POW:PORT1:LEVEL:IMM:AMP 10` sets the power level value to 10 dBm for port 1 on channel 1.

`SOUR2:POW:PORT2:LEVEL:IMM:AMP?` may return the 2, which means 2 dB is the power level value set for port 2 on channel 2.

## SOURce<x>:POWer:SPAN

Set or query the span value of the sweep range for power sweep for the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:SPAN <value>
SOURce<x>:POWer:SPAN?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the span value (dB) for the power sweep for the specified channel.

**Returns** `<NRf>`

**Examples** `SOUR1:POW:SPAN 15` sets the span value for power sweep to 15 dB for channel 1.

`SOUR2:POW:SPAN?` may return 5, which means 5 dB is the span value of the power sweep set for channel 2.

## SOURce<x>:POWer:STARt

Set or query the starting value of the sweep range for power sweep for the specified channel.

**Group** Source commands

**Syntax**

```
SOURce<x>:POWer:STARt <value>
SOURce<x>:POWer:STARt?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the starting value (dBm) of the sweep range for power sweep for the specified channel.

Range: -50 to 10 dBm

**Returns** `<NRf>`

**Examples**

`SOUR1:POW:STARt 5` sets the starting value of the sweep range for power sweep to 5 dBm for channel 1.

`SOUR2:POW:STARt?` may return -3, which means the starting value of the sweep range for power sweep is set to -3 dBm for channel 2.

## SOURce<x>:POWer:STOP

Set or query the stopping value of the sweep range for power sweep for the specified channel.

**Group**

Source commands

**Syntax**

```
SOURce<x>:POWer:STOP <value>
SOURce<x>:POWer:STOP?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NRf>` is the stopping value (dBm) of the sweep range for power sweep for the specified channel.

Range: -50 to 10 dBm

**Returns**

`<NRf>`

**Examples**

`SOUR1:POW:STAR -20` sets the stopping value of the sweep range for power sweep to be -20 dBm for channel 1.

`SOUR2:POW:STAR?` may return -15, which means -15 dBm is the stopping value of the sweep range for power sweep on channel 2.

## SOURce<xz>:POWer:PORT<y>:CORRection:COLLect:AVERage:[COUNt]

Set or query the averaging factor value for power calibration data measurements for the specified port on the specified channel.

**Group**

Source commands

**Syntax**

```
SOURce<xz>:POWer:PORT<y>:CORRection:COLLect:AVERage:[COUNt]
<value>
SOURce<xz>:POWer:PORT<y>:CORRection:COLLect:AVERage:[COUNt]?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16); <y> is the port number (1 or 2).

`<value>::=<NR1>` is the averaging factor value for power calibration data measurements per measurement point.

Range: 1 to 100 dB

**Returns**

`<NR1>`

**Examples**

`SOUR1:POW:PORT1:CORR:COLL:AVER:COUN 6` sets the averaging factor value to 6 dB for port 1 on channel 1.

`SOUR4:POW:PORT2:CORR:COLL:AVER:COUN?` may return 7, which means 7 dB is the averaging factor value set for port 2 on channel 4.

## SOURce<zx>:POWer:[LEVel]:SLOPe:[STATe]

Set or query the status of the power slope for the specified channel.

**Group**

Source commands

**Syntax**

```
SOURce<zx>:POWer:[LEVel]:SLOPe:[STATe] <value>
SOURce<zx>:POWer:[LEVel]:SLOPe:[STATe]?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<boolean>`

**Returns**

1 means the power slope is enabled.

0 means the power slope is disabled.

**Examples**

`SOUR:POW:LEV:SLOP:STAT 1` enables the power slope for channel 1.

`SOUR2:POW:LEV:SLOP:STAT?` may return 0, which means the power slope is disabled for channel 2.

## *SRE

Sets or queries the value of the Service Request Enable Register (SRER). Refer to Section 3 in *Status and Events* for the register information.

**Group** IEEE common commands

**Syntax**

```
*SRE <value>
*SRE?
```

**Related Commands**

*WAI

*ESE

*ESR?

*STB?

**Arguments** `<value>::=<NR1>` is the value of the Service Request Enable Register.

Range: 0 to 255

***NOTE.*** *You must configure binary bits of SRER according to the <NR1> value. Using out-of-range values may result in an executing error.*

**Examples** `*SRE 48` sets binary 00110000 in SRER's bits.

`*SRE?` may return 32, indicating that the binary value 00100000 is set in SRER's bits.

## STATus:OPERation:CONDition? (Query Only)

Query the value of the Operation Status Condition Register.

**Group** Status commands

**Syntax**

```
STATus:OPERation:CONDition?
```

**Returns** `<NR1>`

**Examples** `STAT:OPER:COND?` may return 16, which is the value of the Operation Status Condition Register, if measurement is activated.

## STATus:OPERation:ENABle

Set or query the value of the Operation Status Enable Register.

**Group** Status commands

**Syntax**

```
STATus:OPERation:ENABle <value>
STATus:OPERation:ENABle?
```

**Arguments** `<value>::=<NR1>` is the value of the Operation Status Enable Register.

Range: 0 to 65535

**Returns** `<NR1>`

**Examples** `STAT:OPER:ENAB 16` sets the value of the Operation Status Enable Register to 16.

`STAT:OPER:ENAB?` may return 16, which is the value of the Operation Status Enable Register.

## STATus:OPERation:[EVENt]? (Query Only)

Query the value of the Operation Status Event Register.

**Group** Status commands

**Syntax**

```
STATus:OPERation:[EVENt]?
```

**Returns** `<NR1>`

**Examples** `STAT:OPER:EVE?` may return 0, which is the value of the Operation Status Event Register.

## STATus:OPERation:NTRansition

Set or query the value of the negative transition filter of the Operation Status Register.

**Group** Status commands

**Syntax**

```
STATus:OPERation:NTRansition <value>
STATus:OPERation:NTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the negative transition filter.

Range: 0 to 65535

**Returns** `<NR1>`

**Examples** `STAT:OPER:NTR 16` sets the value of the negative transition filter of the Operation Status Register to 16.

`STAT:OPER:NTR?` may return 16, which is the value of the negative transition filter of the Operation Status Register.

## STATus:OPERation:PTRansition

Set or query the value of the positive transition filter of the Operation Status Register.

**Group** Status commands

**Syntax**

```
STATus:OPERation:PTRansition <value>
STATus:OPERation:PTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the positive transition filter.

Range: 0 to 65535

**Returns** `<NR1>`

**Examples** `STAT:OPER:PTR 16` sets the value of the positive transition filter of the Operation Status Register to 16.

`STAT:OPER:PTR?` may return 16, which is the value of the positive transition filter of the Operation Status Register.

## STATus:PRESet (No Query Form)

Initialize the Status Registers.

**Group** Status commands

**Syntax** `STATus:PRESet`

**Arguments** None

**Examples** `STAT:PRES` initializes all Status Registers.

## STATus:QUEStionable:BLIMit:CHANnel<x>:CONDition? (Query Only)

Query the value of the Questionable Bandwidth Limit Channel Status Condition Register for the specified channel. The register represents the pass/fail results from the bandwidth limit test for traces 1-14 for the specified channel.

**Group** Status commands

**Syntax** `STATus:QUEStionable:BLIMit:CHANnel<x>:CONDition?`

**Related Commands** STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

The pass/fail status for all traces for the specified channel where the bandwidth limit test is enabled:

Bits 0-15 represent the pass/fail status for traces 1-14, where bits 0 and 15 are always 0 and bit 1-14 respectively represent traces 1-14.

If the respective bit value returns 1, it means the bandwidth limit test failed.

If the respective bit value returns 0, it means the bandwidth limit test passed or no test has occurred.

If traces 1-14 are active, the bandwidth limit test is failing for each of these traces.

**Examples** `STAT:QUES:BLIM:CHAN3:COND?` may return 32766 (0111 1111 1111 1110), which is the value of the Questionable Bandwidth Limit Channel Status Condition Register for channel 3, if traces 1-14 are active and the bandwidth limit test is failing for each of these traces.

If trace 1 fails, the return value may be 2 (0000 0000 0000 0010); if trace 14 fails, the return may be 16384 (0100 0000 0000 0000).

## STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition? (Query Only)

Query the value of the Questionable Bandwidth Limit Channel Extra Status Condition Register for the specified channel. The register represents the pass/fail results from the bandwidth limit test for traces 15-16 for the specified channel.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition?
```

**Related Commands** STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

```
<NR1>
```

The pass/fail status for all traces for the specified channel where the bandwidth limit test is enabled.

Bits 0-15 represent the pass/fail status for traces 15 and 16, where bits 2-15 are always 0 and bits 0-1 respectively represent traces 15 and 16.

If the respective bit value returns 1, it means the bandwidth limit test failed.

If the respective bit value returns 0, it means the bandwidth limit test passed or no test has occurred.

**Examples** `STAT:QUES:BLIM:CHAN2:ECH:COND?` may return 3 (0000 0000 0000 0011), which is the value of the Questionable Bandwidth Limit Channel Extra Status Condition Register for channel 2, if traces 15-16 are active and the bandwidth limit test is failing for each of these traces.

## STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:ENABle

Set or query the value of the Questionable Bandwidth Limit Channel Extra Status Enable Register for the specified channel. This enables the corresponding status event register based on the type of transition used for the respective event bit for traces 15-16. Bits used are 0-1, while bits 2 and 15 are always 0.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:ENABle
<value>
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:ENABle?
```

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Bandwidth Limit Channel Extra Status Enable Register.

**Returns**

`<NR1>`

**Examples**

The following command sequence enable the bit corresponding to trace 16 to look for a negative transition. If the bandwidth limit test result changes from fail (1) to pass (0), the query may return 2, indicating that the transition occurred on channel 1 for trace 16:

```
STAT:QUES:BLIM:CHAN1:ECH:ENAB 2
STAT:QUES:BLIM:CHAN1:ECH:NTR 2
STAT:QUES:BLIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:[EVENt]? (Query Only)

Query the value of the Questionable Bandwidth Limit Channel Extra Status Event Register for the specified channel. For traces 15-16, bits 0-1 use 0 for no event occurred and 1 for an event based on the transition type used for the bandwidth limit test.

**Group**

Status commands

**Syntax**

`STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:[EVENt]?`

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition?

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** The following commands enable the bit corresponding to trace 15 to look for a positive transition. If the bandwidth limit test result changes from pass (0) to fail (1), the query may return 1, indicating that the transition occurred on channel 1 for trace 15:

```
STAT:QUES:BLIM:CHAN1:ECH:PTR 1
STAT:QUES:BLIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition

Set or query the value of the negative transition filter of the Questionable Bandwidth Limit Channel Extra Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). You must only use traces 15-16 to represent bits 0-1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition
<value>
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition?
```

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

<value>::=<NR1> is the value of the negative transition filter of the Questionable Bandwidth Limit Channel Extra Status Register.

**Returns** <NR1>

**Examples** The following commands enable the bits corresponding to traces 15 and 16 to look for a negative transition. If the bandwidth limit test result changes from fail (1) to pass (0), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:BLIM:CHAN1:ECH:PTR 3
STAT:QUES:BLIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition

Set or query the value of the positive transition filter of the Questionable Bandwidth Limit Channel Extra Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). You must only use traces 15-16 to represent bits 0-1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition
<value>
STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:PTRansition?
```

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:ECHannel:CONDition?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the positive transition filter of the Questionable Bandwidth Limit Channel Extra Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 15-16 to look for a positive transition. If the bandwidth limit test changes from pass (0) to fail (1), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:BLIM:CHAN1:ECH:PTR 3
STAT:QUES:BLIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:ENABle

Set or query the value of the Questionable Bandwidth Limit Channel Extra Enable Register for the specified channel. This enables the corresponding status event register, based on the type of transition used for the respective event bit for traces 1-14. Bits used are 1-14, where bits 0 and 15 are always 0.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:ENABle <value>
STATus:QUEStionable:BLIMit:CHANnel<x>:ENABle?
```

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Bandwidth Limit Channel Status Enable Register.

**Returns**

`<NR1>`

**Examples**

The following command sequence enable the bits corresponding to traces 1, 2, and 3 to look for a negative transition. If the bandwidth limit test result changes from fail (1) to pass (0) for all three enabled traces, the query may return 14, indicating that the transition occurred on channel 1:

```
STAT:QUES:BLIM:CHAN1:ENAB 14
STAT:QUES:BLIM:CHAN1:NTR 14
STAT:QUES:BLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:[EVENt]? (Query Only)

Query the value of the Questionable Bandwidth Limit Channel Status Event Register for the specified channel. For traces 0-14, bits 0 and 1 use 0 for no event occurred and 1 for an event based on the transition type used for the bandwidth limit test.

**Group**

Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:[EVENt]?
```

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:CONDition?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

<NR1>

**Examples** The following commands enable the bits corresponding to traces 15-16 to look for a positive transition. If the bandwidth limit test result changes from pass (0) to fail (1), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:BLIM:CHAN1:PTR 112
STAT:QUES:BLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition

Set or query the value of the negative transition filter of the Questionable Bandwidth Limit Channel Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used to represent bits 1-14.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition <value>
STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition?
```

**Related Commands** STATus:QUEStionable:BLIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

<value>::=<NR1> is the value of the negative transition filter of the Questionable Bandwidth Limit Channel Status Register.

**Returns** <NR1>

**Examples** The following commands enable the bits corresponding to traces 7, 8, 9 and 10 to look for a negative transition. If the bandwidth limit test result changes from fail (1) to pass (0) for all four enabled traces, the query may return 1920, indicating that the transition occurred on channel 1:

```
STAT:QUES:BLIM:CHAN1:NTR 1920
STAT:QUES:BLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition

Set or query the value of the positive transition filter of the Questionable Bandwidth Limit Channel Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the bandwidth limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used to represent bits 1-14.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition <value>
STATus:QUEStionable:BLIMit:CHANnel<x>:PTRansition?
```

**Related Commands**

STATus:QUEStionable:BLIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:BLIMit:CHANnel<x>:CONDition?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the positive transition filter of the Questionable Bandwidth Limit Channel Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 11, 12, 13 and 14 to look for a positive transition. If the bandwidth limit test result changes from pass (0) to fail (1) for all four enabled traces, the query may return 30720, indicating that the transition occurred on channel 1:

```
STAT:QUES:BLIM:CHAN1:PTR 30720
STAT:QUES:BLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:BLIMit:CONDition? (Query Only)

Query the value of the Questionable Bandwidth Limit Status Condition Register. For channels 1-14, bits 1-14 are defined by 0 for pass and 1 for fail. Bit 0 ORs the enabled bits of the extra status condition register.

**Group** Status commands

**Syntax** `STATus:QUEStionable:BLIMit:CONDition?`

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** `STAT:QUES:BLIM:COND?` may return 14, which is the value of the Questionable Bandwidth Limit Status Condition Register.

## STATus:QUEStionable:BLIMit:ENABle

Set or query the value of the Questionable Limit Status Enable Register. The preset value for the register varies depending on the upper limit setting for the channel/trace. You cannot set bit 15 of this register to 1.

**Group** Status commands

**Syntax**
```
STATus:QUEStionable:BLIMit:ENABle <value>
STATus:QUEStionable:BLIMit:ENABle?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Limit Status Enable Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:BLIM:ENAB 20` sets the value of the Questionable Limit Status Enable Register to 20.

`STAT:QUES:BLIM:ENAB?` may return 20, which is the value of the Questionable Limit Status Enable Register.

## STATus:QUEStionable:BLIMit:[EVENt]? (Query Only)

Query the value of the Questionable Limit Status Event Register.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:BLIMit:[EVENt]?
```

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

```
<NR1>
```

**Examples**

`STAT:QUES:BLIM:EVEN?` may return 16 as the value of the Questionable Limit Status Event Register.

## STATus:QUEStionable:LIMit:CHANnel<x>:CONDition? (Query Only)

Query the value of the Questionable Limit Channel Status Condition Register for the specified channel. The register represents the pass/fail results from the limit test for traces 1-14 for the specified channel.

**Group**

Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:CONDition?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition?

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

```
<NR1>
```

The pass/fail status for all traces for the specified channel, where the limit test is enabled:

Bits 0-15 represent the pass/fail status for traces 1-14, where bits 0 and 15 are always 0, while bits 1-14 respectively represents traces 1-14.

If the respective bit value returns 1, limit test has failed, while 0 means no test has occurred or that it passed the limit test.

**Examples**

`STAT:QUES:LIM:CHAN3:COND?` may return 20 (0000 0000 0000 1010), which is the value of the Questionable Limit Channel Status Condition Register set for channel 3, if traces 2 and 4 are active and the limit test is failing for each of these traces.

## STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition? (Query Only)

Query the value of the Questionable Limit Channel Extra Status Condition Register for the specified channel. The register represents the pass/fail result from the limit test for traces 1-14 for the specified channel.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition?
```

**Related Commands** STATus:QUEStionable:LIMit:CHANnel<x>:CONDition?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

The pass/fail status for all traces for the specified channel, where the limit test is enabled:

Bits 0-15 represent the pass/fail status for traces 1-14, where bits 0 and 15 are always , while bits 1-14 respectively represent traces 1-14.

If the respective bit value returns 1, limit test has failed, while 0 means no test has occurred or that it passed the limit test.

**Examples** `STAT:QUES:LIM:CHAN1:ECH:COND?` may return 15, which is the value of the Questionable Limit Channel Extra Status Condition Register set for channel 1, if traces 15-16 are active and the limit test is enabled and failing for each of these traces.

## STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:ENABle

Set or query the value of the Questionable Limit Channel Extra Status Enable Register for the specified channel. This command enables the corresponding status event register based on the type of transition used for the respective event bit for traces 15-16. Bits used are 0-1, while bits 2 and 15 are always 0.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:ENABle <value>
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:ENABle?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Limit Channel Extra Status Enable Register.

**Returns**

`<NR1>`

**Examples**

The following commands enable the bit corresponding to trace 16 to look for a negative transition. If the limit test result changes from fail (1) to pass (0), the query then returns 2, which is the value indicating that the transition occurred on channel 1 for trace 16:

```
STAT:QUES:LIM:CHAN1:ECH:ENAB 2
STAT:QUES:LIM:CHAN1:ECH:NTR 2
STAT:QUES:LIM:CHAN1:ECH:ENAB?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:[EVENt]? (Query Only)

Query the value of the Questionable Limit Channel Extra Status Event Register for the specified channel. For traces 15-16, bits 0-1 indicate 0 for no event occurred and 1 for an event based on the transition type used for limit test.

**Group**

Status commands

**Syntax**

`STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:[EVENt]?`

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition?

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** The following commands enable the bit corresponding to trace 15 to look for a positive transition. If the limit test result changes from pass (0) to fail (1), the query may return 1, indicating that the transition occurred on channel 1 for trace 15:

```
STAT:QUES:LIM:CHAN1:ECH:PTR 1
STAT:QUES:LIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition

Set or query the value of the negative transition filter of the Questionable Limit Channel Extra Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the limit test, when the test result transitions from fail (1) to pass (0). Traces 15-16 are only used to represent bits 0-1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition
<value>
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the negative transition filter of the Questionable Limit Channel Extra Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 15 and 16 to look for a negative transition. If the limit test result changes from fail (1) to pass (0), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:LIM:CHAN1:ECH:NTR 3
STAT:QUES:LIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition

Set or query the value of the positive transition filter of the Questionable Limit Channel Extra Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the limit test, when the test result transitions from pass (0) to fail (1). Traces 15-16 are only used to represent bits 0-1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition
<value>
STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:PTRansition?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:ECHannel:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the positive transition filter of the Questionable Limit Channel Extra Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 15 and 16 to look for a negative transition. If the limit test result changes from fail (1) to pass (0), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:LIM:CHAN1:ECH:PTR 3
STAT:QUES:LIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:ENABle

Set or query the value of the Questionable Limit Channel Status Enable Register for the specified channel. This enables the corresponding status event register based on the type of transition used for the respective event bit for traces 1-14. Bits used are 1-14, where bits 0 and 15 are always 0.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:ENABle <value>
STATus:QUEStionable:LIMit:CHANnel<x>:ENABle?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:BLIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Limit Channel Status Enable Register.

**Returns**

`<value>`

**Examples**

The following commands enable the bits corresponding to traces 1, 2, and 3 to look for a negative transition. If the limit test result changes from fail (1) to pass (0) for all three enabled traces, the query may return 14, indicating that the transition occurred on channel 1:

```
STAT:QUES:LIM:CHAN1:ENAB 14
STAT:QUES:LIM:CHAN1:NTR 14
STAT:QUES:LIM:CHAN1:ENAB?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:[EVENt]? (Query Only)

Query the value of the Questionable Limit Channel Status Event Register for the specified channel.

**Group**

Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:[EVENt]?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:CONDition?

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

<NR1>

**Examples** The following commands enable the bits corresponding to traces 4, 5, and 6 to look for a positive transition. If the limit test result changes from pass (0) to fail (1) for all three enabled traces, the query may return 112, indicating that the transition occurred on channel 1:

```
STAT:QUES:LIM:CHAN1:PTR 112
STAT:QUES:LIM:CHAN1:EVEN?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:NTRansition

Set or query the value of the negative transition filter of the Questionable Limit Channel Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the limit test, when the test result transitions from fail (1) to pass (0). Traces 1-14 are only used to represent bits 1-14.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:NTRansition <value>
STATus:QUEStionable:LIMit:CHANnel<x>:NTRansition?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

<value>::=<NR1> is the value of the negative transition filter of the Questionable Limit Channel Status Register.

**Returns** <NR1>

**Examples** The following commands enable the bits corresponding to traces 7, 8, 9 and 10 to look for a negative transition. If the limit test result changes from fail (1) to pass (0) for all four enabled traces, the query may return 1920, indicating that the transition occurred on channel 1:

```
STAT:QUES:LIM:CHAN1:NTR 1920
```

```
STAT:QUES:LIM:CHAN1:EVEN?
```

## STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition

Set or query the value of the positive transition filter of the Questionable Limit Channel Status Register for the specified channel. A positive transition is defined when the condition register changes from 0 to 1. This occurs during the limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used to represent bits 1-14.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition <value>
STATus:QUEStionable:LIMit:CHANnel<x>:PTRansition?
```

**Related Commands**

STATus:QUEStionable:LIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:LIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:LIMit:CHANnel<x>:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the positive transition filter of the Questionable Limit Channel Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 11, 12, 13 and 14 to look for a positive transition. If the limit test result changes from pass (0) to fail (1) for all four enabled traces, the query may return 30720, indicating that the transition occurred on channel 1:

```
STAT:QUES:LIM:CHAN1:PTR 30720
STAT:QUES:LIM:CHAN1:EVEN?
```

## STATus:QUEStionable:LIMit:CONDition? (Query Only)

Query the value of the Questionable Limit Status Condition Register.

**Group** Status commands

**Syntax** `STATus:QUEStionable:LIMit:CONDition?`

**Returns** `<NR1>`

**Examples** `STAT:QUES:LIM:COND?` may return 15, which is the value of the Questionable Limit Status Condition Register.

## STATus:QUEStionable:LIMit:ELIMit:CONDition? (Query Only)

Query the value of the Questionable Limit Extra Status Condition Register.

**Group** Status commands

**Syntax** `STATus:QUEStionable:LIMit:ELIMit:CONDition?`

**Returns** `<NR1>`

**Examples** `STAT:QUES:LIM:ELIM:COND?` may return 15, which is the value of the Questionable Limit Extra Status Condition Register.

## STATus:QUEStionable:LIMit:ELIMit:ENABle

Set or query the value of the Questionable Limit Extra Status Enable Register.

**Group** Status commands

**Syntax**
```
STATus:QUEStionable:LIMit:ELIMit:ENABle <value>
STATus:QUEStionable:LIMit:ELIMit:ENABle?
```

**Arguments** `<value>::=<NR1>` is the value of the Questionable Limit Extra Status Enable Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:LIM:ELIM:ENAB?` may return 5, which is the value of the Questionable Limit Extra Status Enable Register.

`STAT:QUES:LIM:ELIM:ENAB 2` sets the value of the Questionable Limit Extra Status Enable Register to 2.

## STATus:QUEStionable:LIMit:ELIMit:EVENt? (Query Only)

Query the value of the Questionable Limit Extra Status Event Register.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:ELIMit:EVENt?
```

**Returns**

```
<NR1>
```

**Examples** `STAT:QUES:LIM:ELIM:EVEN?` may return 4 as the value of the Questionable Limit Extra Status Event Register.

## STATus:QUEStionable:LIMit:ELIMit:NTRansition

Set or query the value of value of the negative transition filter of the Questionable Limit Extra Status Register. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:ELIMit:NTRansition <value>
STATus:QUEStionable:LIMit:ELIMit:NTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the negative transition filter of the Questionable Limit Extra Status Register.

**Returns**

```
<NR1>
```

**Examples** `STAT:QUES:LIM:ELIM:NTR 8` sets the value of the negative transition filter of the Questionable Limit Extra Status Register to 8.

`STAT:QUES:LIM:ELIM:NTR?` may return 4, which is the value of the negative transition filter of the Questionable Limit Extra Status Register.

## STATus:QUEStionable:LIMit:ELIMit:PTRansition

Set or query the value of the positive transition filter of the Questionable Limit Extra Status Register. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:ELIMit:PTRansition <value>
STATus:QUEStionable:LIMit:ELIMit:PTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the positive transition filter of the Questionable Limit Extra Status Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:LIM:ELIM:PTR 6` sets the value of the positive transition filter of the Questionable Limit Extra Status Register to 6.

`STAT:QUES:LIM:ELIM:PTR?` may return 2, which is the value of the positive transition filter of the Questionable Limit Extra Status Register.

## STATus:QUEStionable:LIMit:ENABle

Set or query the value of the Questionable Limit Status Enable Register.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:ENABle <value>
STATus:QUEStionable:LIMit:ENABle?
```

**Arguments** `<value>::=<NR1>` is the value of the Questionable Limit Status Enable Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:LIM:ENAB?` may return 5, which is the value of the Questionable Limit Status Enable Register.

`STAT:QUES:LIM:ENAB 2` sets the value of the Questionable Limit Status Enable Register to 2.

## STATus:QUEStionable:LIMit:[EVENt]? (Query Only)

Query the value of the Questionable Limit Status Event Register.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:[EVENt]?
```

**Returns**

```
<NR1>
```

**Examples** `STAT:QUES:LIM:EVEN?` may return 4, which is the value of the Questionable Limit Status Event Register.

## STATus:QUEStionable:LIMit:NTRansition

Set or query the value of value of the negative transition filter of the Questionable Limit Status Register. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:NTRansition <value>
STATus:QUEStionable:LIMit:NTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the negative transition filter of the Questionable Limit Status Register.

**Returns**

```
<NR1>
```

**Examples** `STAT:QUES:LIM:NTR 8` sets the value of the negative transition filter of the Questionable Limit Status Register to 8.

`STAT:QUES:LIM:NTR?` may return 5, which is the value of the negative transition filter of the Questionable Limit Status Register.

## STATus:QUEStionable:LIMit:PTRansition

Set or query the value of the positive transition filter of the Questionable Limit Status Register for the specified channel. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:LIMit:PTRansition <value>
STATus:QUEStionable:LIMit:PTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the positive transition filter of the Questionable Limit Status Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:LIM:PTR 6` sets the value of the positive transition filter of the Questionable Limit Status Register to 6.

`STAT:QUES:LIM:PTR?` may return 2, which is the value of the positive transition filter of the Questionable Limit Status Register.

## STATus:QUEStionable:NTRansition

Set or query the value of the negative transition filter of the Questionable Status Register. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:NTRansition <value>
STATus:QUEStionable:NTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the negative transition filter of the Questionable Status Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:NTR 8` sets the value of the negative transition filter of the Questionable Status Register to 8.

`STAT:QUES:NTR?` may return 3, which is the value of the negative transition filter of the Questionable Status Register.

## STATus:QUEStionable:PTRansition

Set or query the value of the positive transition filter of the Questionable Status Register for the specified channel. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:PTRansition <value>
STATus:QUEStionable:PTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the positive transition filter of the Questionable Status Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:PTR 6` sets the value of the positive transition filter of the Questionable Status Register to 6.

`STAT:QUES:PTR?` may return 1, which is the value of the positive transition filter of the Questionable Status Register.

## STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition? (Query Only)

Query the value of the Questionable Ripple Limit Channel Status Condition Register for the specified channel. The register represents the pass/fail results from the ripple limit test for traces 1-14 on the specified channel.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition?
```

**Related Commands** STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:CONDition?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

The pass/fail status for all traces for the specified channel, where the ripple limit test is enabled:

Bits 0-15 represent the pass/fail status for traces 1-14, where bits 0 and 15 are always 0, and bit 1-14 respectively represents traces 1-14.

If the respective bit value returns 1, the bandwidth limit test failed.

If the respective bit value returns, it means that either no test has occurred or that it passed the ripple limit test.

**Examples** `STAT:QUES:RLIM:CHAN1:COND?` may return 14 (0000 0000 0000 1110), which is the value of the Questionable Ripple Limit Channel Status Condition Register set for channel 1, if traces 1, 2, and 3 are active and the ripple limit test is failing for each of these traces.

## STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:CONDition? (Query Only)

Query the value of the Questionable Ripple Limit Channel Extra Status Condition Register for the specified channel. The register represents the pass/fail results from the ripple limit test results for traces 15-16 on the specified channel.

**Group** Status commands

**Syntax** `STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:CONDition?`

**Related Commands** STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition?

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

The pass/fail status for all traces for the specified channel where the ripple limit test is enabled:

Bits 0-15 represent the pass/fail status for traces 15 and 16, where bits 2-15 are always 0 and bits 0-1 respectively represent traces 15 and 16.

If the respective bit value returns 1, the Ripple Limit Test has failed.

If the respective bit value returns 0, it means that no test has occurred or that it passed the ripple limit test.

**Examples** `STAT:QUES:RLIM:CHAN1:ECH:COND?` may return 2, which is the value of the Questionable Ripple Limit Channel Extra Status Condition Register set for channel 1, if trace 16 is active and the ripple limit test is failing for this trace.

## STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:ENABle

Set or query the value of the Questionable Ripple Limit Channel Extra Status Enable Register for the specified channel.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:ENABle
<value>
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:ENABle?
```

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Ripple Limit Channel Extra Status Enable Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:RLIM:CHAN1:ECH:ENAB?` may return 5, which is the value of the Questionable Ripple Limit Channel Extra Status Enable Register set for channel 1.

`STAT:QUES:RLIM:CHAN2:ECH:ENAB 3` sets 3 to be the value of the Questionable Ripple Limit Channel Extra Status Enable Register for channel 2.

## STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:[EVENt]? (Query Only)

Query the value of the Questionable Ripple Limit Channel Extra Status Event Register for the specified channel. For traces 15-16, the bits 0-1 indicate 0 for no event occurred and 1 for an event based on the transition type used for the ripple limit test.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:[EVENt]?
```

**Related Commands** STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:PTRansition

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** The following commands enable the bit corresponding to trace 15 to look for a positive transition. If the ripple limit test result changes from pass (0) to fail (1), the query may return 1, indicating that the transition occurred for trace 15 on channel 1:

```
STAT:QUES:RLIM:CHAN1:ECH:PTR 1
STAT:QUES:RLIM:CHAN2:ECH:EVEN?
```

## STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:NTRansition

Set or query the value of the negative transition filter of the Questionable Ripple Limit Channel Extra Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the ripple limit test, when the test result transitions from fail (1) to pass (0). Traces 15-16 are only used, representing bits 0-1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:NTRansition
<value>
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:NTRansition?
```

**Related Commands**

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:PTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

<value>::=<NR1> is the value of the negative transition filter of the Questionable Ripple Limit Channel Extra Status Register.

**Returns** <NR1>

**Examples** The following commands enable the bits corresponding to traces 15 and 16 to look for a negative transition. If the ripple limit test result changes from fail (1) to pass (0), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:RLIM:CHAN1:ECH:NTR 3
STAT:QUES:RLIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:PTRansition

Set or query the value of the positive transition filter of the Questionable Ripple Limit Channel Extra Status Register for the specified channel. A positive transition is defined when the condition register changes from a 0 to a 1. This occurs during the ripple limit test, when the test result transitions from pass (0) to fail (1). Traces 15-16 are only used, representing bits 0-1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:PTRansition
<value>
STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:PTRansition?
```

**Related Commands**

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:[EVENt]?

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:NTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:ECHannel:CONDition?

**Arguments** **Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the positive transition filter of the Questionable Ripple Limit Channel Extra Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 15-16 to look for a positive transition. If the ripple limit test result changes from pass (0) to fail (1), the query may return 1, 2, or 3, depending on which traces saw the transition occurred on channel 1:

```
STAT:QUES:RLIM:CHAN1:ECH:PTR 3
STAT:QUES:RLIM:CHAN1:ECH:EVEN?
```

## STATus:QUEStionable:RLIMit:CHANnel<x>:ENABle

Set or query the value of the Questionable Ripple Limit Channel Status Enable Register for the specified channel.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:ENABle <value>
STATus:QUEStionable:RLIMit:CHANnel<x>:ENABle?
```

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the Questionable Ripple Limit Channel Status Enable Register.

**Returns**

`<NR1>`

**Examples**

`STAT:QUES:RLIM:CHAN1:ENAB?` may return 5, which is the value of the Questionable Ripple Limit Channel Status Enable Register for channel 1.

`STAT:QUES:RLIM:CHAN6:ENAB 2` sets the value of the Questionable Ripple Limit Channel Status Enable Register to 2 for channel 6.

## STATus:QUEStionable:RLIMit:CHANnel<x>:[EVENt]? (Query Only)

Query the value of the Questionable Ripple Limit Channel Status Event Register for the specified channel.

**Group**

Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:[EVENt]?
```

**Related Commands**

STATus:QUEStionable:RLIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:PTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition?

**Returns**

**Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples**

The following commands enable the bits corresponding to traces 4, 5, and 6 to look for a positive transition. If the ripple limit test result changes from pass (0) to fail (1) for all three enabled traces, the query may return 112, indicating that the transition occurred on channel 1:

```
STAT:QUES:RLIM:CHAN1:PTR 112
STAT:QUES:RLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:RLIMit:CHANnel<x>:NTRansition

Set or query the value of the negative transition filter of the Questionable Ripple Limit Channel Status Register for the specified channel. A negative transition is defined when the condition register changes from 1 to 0. This occurs during the ripple limit test, when the test result transitions from fail (1) to pass (0). Traces 1-14 are only used, representing bits 1-14.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:NTRansition <value>
STATus:QUEStionable:RLIMit:CHANnel<x>:NTRansition?
```

**Related Commands**

STATus:QUEStionable:RLIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:RLIMit:CHANnel<x>:PTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the negative transition filter of the Questionable Ripple Limit Channel Status Register.

**Returns** `<NR1>`

**Examples** The following commands enable the bits corresponding to traces 7, 8, 9 and 10 to look for a negative transition. If the ripple limit test result changes from fail (1) to pass (0) for all four enabled traces, the query may return 1920, indicating that the transition occurred on channel 1:

```
STAT:QUES:RLIM:CHAN1:NTR 1920
STAT:QUES:RLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:RLIMit:CHANnel<x>:PTRansition

Set or query the value of the positive transition filter of the Questionable Ripple Limit Channel Status Register for the specified channel. A positive transition is defined when the condition register changes from a 0 to a 1. This occurs during the ripple limit test, when the test result transitions from pass (0) to fail (1). Traces 1-14 are only used, representing bits 1-14.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CHANnel<x>:PTRansition <value>
STATus:QUEStionable:RLIMit:CHANnel<x>:PTRansition?
```

**Related Commands**

STATus:QUEStionable:RLIMit:CHANnel<x>:[EVENt]?

STATus:QUEStionable:RLIMit:CHANnel<x>:NTRansition

STATus:QUEStionable:RLIMit:CHANnel<x>:CONDition?

**Arguments**

**Command variables.** <x> is the channel number (1 to 16).

`<value>::=<NR1>` is the value of the positive transition filter of the Questionable Ripple Limit Channel Status Register.

**Returns**

`<NR1>`

**Examples**

The following commands enable the bits corresponding to traces 11, 12, 13 and 14 to look for a positive transition. If the ripple limit test result changes from pass (0) to fail (1) for all four enabled traces, the query may return 30720, indicating that the transition occurred on channel 1:

```
STAT:QUES:RLIM:CHAN1:PTR 30720
STAT:QUES:RLIM:CHAN1:EVEN?
```

## STATus:QUEStionable:RLIMit:CONDition? (Query Only)

Query the value of the Questionable Ripple Limit Status Condition Register.

**Group**

Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:CONDition?
```

**Returns**

`<NR1>`

**Examples**

`STAT:QUES:RLIM:COND?` may return 15, which is the value of the Questionable Ripple Limit Status Condition Register.

## STATus:QUEStionable:RLIMit:ELIMit:CONDition? (Query Only)

Query the value of the Questionable Ripple Limit Extra Status Condition Register.

**Group** Status commands

**Syntax** `STATus:QUEStionable:RLIMit:ELIMit:CONDition?`

**Returns** **Command variables.** <x> is the channel number (1 to 16).

`<NR1>`

**Examples** `STAT:QUES:RLIM:ELIM:COND?` may return 15, which is the value of the Questionable Ripple Limit Extra Status Condition Register.

## STATus:QUEStionable:RLIMit:ENABle

Set or query the value of the Questionable Ripple Limit Status Enable Register.

**Group** Status commands

**Syntax**
```
STATus:QUEStionable:RLIMit:ENABle <value>
STATus:QUEStionable:RLIMit:ENABle?
```

**Arguments** `<value>::=<NR1>` is the value of the positive transition filter of the Questionable Ripple Limit Status Enable Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:RLIM:ENAB 8` sets the value of the Questionable Ripple Limit Status Enable Register to 8.

`STAT:QUES:RLIM:ENAB?` may return 5, which is the value of the Questionable Ripple Limit Status Enable Register.

## STATus:QUEStionable:RLIMit:[EVENt]? (Query Only)

Set or query the value of the Questionable Ripple Limit Status Event Register.

**Group** Status commands

**Syntax** `STATus:QUEStionable:RLIMit:[EVENt]?`

**Returns** `<NR1>`

**Examples** `STAT:QUES:RLIM:EVEN?` may return 4, which is the value of the Questionable Ripple Limit Status Event Register.

## STATus:QUEStionable:RLIMit:NTRansition

Set or query the value of the negative transition filter of the Questionable Ripple Limit Status Register. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:NTRansition
STATus:QUEStionable:RLIMit:NTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the negative transition filter of the Questionable Ripple Limit Status Register.

**Returns** `<NR1>`

**Examples** `STAT:QUES:RLIM:NTR 8` sets the value of the negative transition filter of the Questionable Ripple Limit Status Register to 8.

## STATus:QUEStionable:RLIMit:PTRansition

Set or query the value of the positive transition filter of the Questionable Ripple Limit Status Register. You cannot set bits 0 and 3-15 to 1.

**Group** Status commands

**Syntax**

```
STATus:QUEStionable:RLIMit:PTRansition
STATus:QUEStionable:RLIMit:PTRansition?
```

**Arguments** `<value>::=<NR1>` is the value of the positive transition filter of the Questionable Ripple Limit Status Register.

**Returns** `<NR1>`

**Examples**

`STAT:QUES:RLIM:PTR 6` sets the value of the positive transition filter of the Questionable Ripple Limit Status Register to 6.

`STAT:QUES:RLIM:PTR?` may return 1, which is the value of the positive transition filter of the Questionable Ripple Limit Status Register.

## *STB? (Query Only)

Query the contents of the Status Byte Register (SBR) in the status/event reporting structure, using the Master Summary Status (MSS) bit. Refer to *Status and Events*, Section 3, for the register information.

**Group**

IEEE common commands

**Syntax**

`*STB?`

**Related Commands**

*WAI

*ESE

*ESR?

*SRE

**Arguments**

None

**Returns**

`<NRf>`

**Examples**

`*STB?` may return 96, indicating that SBR contains the binary 0110 0000.

## SYSTem:BEEPer:WARNing:IMMediate (No Query Form)

Generate an immediate beep to verify if the warning beep works properly.

**Group**

System commands

**Syntax**

`SYSTem:BEEPer:WARNing:IMMediate`

**Arguments**

None

**Examples** `SYST:BEEP:WARN:IMM` generates an immediate beep.

## SYSTem:BEEPer:WARNing:[STATe]

Set or query the status of the beeper, which indicates a warning, error or an out of range limit notifications.

**Group** System commands

**Syntax**

```
SYSTem:BEEPer:WARNing:[STATe] <value>
SYSTem:BEEPer:WARNing:[STATe]?
```

**Arguments** `<value>::=<boolean>`

**Returns** 1 means the beeper is turned on.

0 means the beeper is turned off.

**Examples** `SYST:BEEP:WARN:STAT 1` enables the beeper for warning notifications.

`SYST:BEEP:WARN:STAT?` may return 0, which indicates the beeper is disabled for warning notifications.

## SYSTem:BIAStee

Set or query the state of the bias tee function.

**Group** System commands

**Syntax**

```
SYSTem:BIAStee <value>
SYSTem:BIAStee?
```

**Arguments** `<value>::=<boolean>`

**Returns** 1 means the bias tee function is turned on.

0 means the bias tee function is turned off.

**Examples** `SYST:BIAS 1` enables the bias tee function.

`SYST:BIAS?` may return 0, which means the bias tee function is disabled.

## SYSTem:COMMunicate:VISA:PMETer:ADDRess

Set or query the VISA address of the power meter used with the TTR500 instrument.

**Group** System commands

**Syntax**

```
SYSTem:COMMunicate:VISA:PMETer:ADDRess <value>
SYSTem:COMMunicate:VISA:PMETer:ADDRess?
```

**Arguments** `<value>::=<string>` is the VISA address of the power meter.

**Returns** `<string>`

**Examples** `SYST:COMM:VISA:PMET:ADDR "USB0::0x3923::0x72A1::0125F0B3::RAW"` sets the VISA address of the power meter to USB0::0x3923::0x72A1::0125F0B3::RAW.

`SYST:COMM:VISA:PMET:ADDR?` may return the VISA address of the power mete that is paired with the TTR500 instrument.

## SYSTem:COMMunicate:VISA:PMETer:MODel

Set or display the model name of the power meter used with the TTR500 instrument.

**Group** System commands

**Syntax**

```
SYSTem:COMMunicate:VISA:PMETer:MODel <string>
SYSTem:COMMunicate:VISA:PMETer:MODel?
```

**Arguments** `<string>` is the model name of the power meter. Select one from the following list:

| Power meter model names |
| --- |
| TEKTRONIX_PSM3000 |
| TEKTRONIX_PSM3000 |
| TEKTRONIX_PSM4000 |
| TEKTRONIX_PSM5000 |
| KEYSIGHT_U848X |
| KEYSIGHT_U2020 |
| KEYSIGHT_U2000 |
| RNS_NRP_Z |
| RNS_NRP_XXS_SN |

**Examples**

`SYST:COMM:VISA:PMET:MOD "TEKTRONIX_PSM3000"` pairs Tektronix PSM3000 power meter with the TTR500 instrument.

`SYST:COMM:VISA:PMET:MOD?` may return TEKTRONIX_PSM4000, which is the name of the power meter paired with the TTR500 instrument.

## SYSTem:CONNect

Set or query the name of the connected instrument.

**Group**

System commands

**Syntax**

```
SYSTem:CONNect <value>
SYSTem:CONNect?
```

**Arguments**

`<value>::=<string>` is the name of the connected instrument.

**Examples**

`SYST:CONN "TTR506A_Y010060_C3022822"` connects VectorVu-PC to the specified TTR500 instrument.

`SYST:CONN?` returns the instrument name currently connected to VectorVu-PC.

## SYSTem:CORRection:PERFormance

Set or query the corrected performance measurement.

**Group**

System commands

**Syntax**

```
SYSTem:CORRection:PERFormance <value>
SYSTem:CORRection:PERFormance?
```

**Arguments**

```
<value>::=<boolean>
```

**Returns**

1 means corrected performance measurement is turned on.

0 means corrected performance measurement is turned off.

**Examples**

`SYST:CORR:PERF 1` enables corrected performance measurement.

`SYST:CORR:PERF?` may return 0, which means corrected performance measurement is disabled.

## SYSTem:CORRection:[STATe]

Set or query the status of system error correction.

**Group**

System commands

**Syntax**

```
SYSTem:CORRection:[STATe] <value>
SYSTem:CORRection:[STATe]?
```

**Arguments**

```
<value>::=<boolean>
```

**Returns**

1 means system error correction is turned on.

0 means system error correction is turned off.

**Examples**

`SYST:CORR:STAT 1` enables system error correction.

`SYST:CORR:STAT?` may return 0, which means system error correction is disabled.

## SYSTem:DISConnect (No Query Form)

Disconnect the disaggregate TTR500 instrument.

**Group**

System commands

**Syntax** `SYSTem:DISConnect`

**Arguments** None

**Examples** `SYST:DISC` disconnects the disaggregate TTR500 instrument.

## SYSTem:DISCOVER? (Query Only)

Query the list of disaggregate instruments available on the USB bus and simulator.

**Group** System commands

**Syntax** `SYSTem:DISCOVER?`

**Returns** This command displays a list of the disaggregate instruments available on the USB bus and simulator.

**Examples** `SYST:DISCOVER?` may return a list of the disaggregate instruments available on the USB bus and simulator.

## SYSTem:ERRor:ALL? (Query Only)

Display information about all errors and events.

***NOTE.*** *If this command finds no errors, the return value will be 0.*

**Group** System commands

**Syntax** `SYSTem:ERRor:ALL?`

**Returns** <error code>,<error/event description>,<error/event details>

**Examples** `SYST:ERR:ALL?` may return a list of all errors, descriptions and details about errors or events.

## SYSTem:ERRor:CODE:ALL? (Query Only)

Query the code numbers for all errors and events.

**Group** System commands

**Syntax** `SYSTem:ERRor:CODE:ALL?`

**Returns** The list of all error codes

**Examples** `SYST:ERR:CODE:ALL?` may return a list of all error codes.

## SYSTem:ERRor:CODE:[NEXT]? (Query Only)

Query the next error or event code numbers.

**Group** System commands

**Syntax** `SYSTem:ERRor:CODE:[NEXT]?`

**Returns** The next error or event code.

**Examples** `SYST:ERR:CODE:NEXT?` may return 501, which is the next error code.

## SYSTem:ERRor:COUNt? (Query Only)

Display the total number of errors and events.

**Group** System commands

**Syntax** `SYSTem:ERRor:COUNt?`

**Returns** `<NR1>`

**Examples** `SYST:ERR:COUN?` may return 6, which is the total number of errors and events.

## SYSTem:ERRor:[NEXT]? (Query Only)

Query the error code, description, and details about the next error or event code.

**Group** System commands

**Syntax** `SYSTem:ERRor:[NEXT]?`

**Returns** <error code>,<error/event description>,<error/event details>

**Examples** `SYST:ERR:NEXT?` may return 0, No error, which means no error was found.

## SYSTem:PRESet:MODE

Set or query the preset mode for the TTR500 instrument.

**Group** System commands

**Syntax**

```
SYSTem:PRESet:MODE { FACtory | USER }
SYSTem:PRESet:MODE?
```

**Arguments** `FACtory` sets factory preset mode.

`USER` sets user preset mode.

**Returns** `FACtory` means factory preset mode is set.

`USER` means user preset mode is set.

**Examples** `SYST:PRES:MODE FAC` sets the preset mode to factory.

`SYST:PRES:MODE?` may return USER, which means the preset mode is set to user.

## SYSTem:SERVice? (Query Only)

Query the operating mode (normal or service mode) for the TTR500 instrument.

**Group** System commands

**Syntax**

```
SYSTem:SERVice?
```

**Returns**

1 means the operating mode is service.

0 means the operating mode is normal.

**Examples**

`SYST:SERV?` may return 1, which means the TTR500 instrument is set to service mode.

`SYST:SERV?` may return 0, which means the TTR500 instrument is set to normal mode.

## SYSTem:STACking:TEST

Set or query the state of the stacking test for the TTR500 instrument.

**Group**

System commands

**Syntax**

```
SYSTem:STACking:TEST <value>
SYSTem:STACking:TEST?
```

**Arguments**

```
<value>::=<boolean>
```

**Returns**

1 means the stacking test is turned on.

0 means the stacking test is turned off.

**Examples**

`SYST:STAC:TEST 1` enables the stacking test for the TTR500 instrument.

`SYST:STAC:TEST?` may return 0, which means the stacking test is disabled for the TTR500 instrument.

## SYSTem:STACking:TIMing

Set or query the state of the stacking timing test for the TTR500 instrument.

**Group**

System commands

**Syntax**

```
SYSTem:STACking:TIMing <value>
SYSTem:STACking:TIMing?
```

**Arguments** `<value>::=<boolean>`

**Returns** 1 means the stacking timing test is turned on.

0 means the stacking timing test is turned off.

**Examples** `SYST:STAC:TIM 1` enables the stacking timing test for the TTR500 instrument.

`SYST:STAC:TIM?` may return 0, which means the stacking timing test is disabled for the TTR500 instrument.

## SYSTem:TEMPerature:DATA? (Query Only)

Query the temperature sensor data for the TTR500 instrument.

**Group** System commands

**Syntax** `SYSTem:TEMPerature:DATA?`

**Returns** `<NRf>`

**Examples** `SYST:TEMP:DATA?` may return 52, which means 52 °C (degrees Celsius) is the operating temperature of the TTR500 instrument.

## SYSTem:TEMPerature:[STATe]? (Query Only)

Query the warm up state of the TTR500 instrument.

**Group** System commands

**Syntax** `SYSTem:TEMPerature:[STATe]?`

**Returns** 1 means the TTR500 instrument is warmed up.

0 means the TTR500 instrument is not warmed up.

**Examples** `SYST:TEMP:STAT?` may return 1, which means the TTR500 instrument is warmed up.

`SYST:TEMP:STAT?` may return 0, which means the TTR500 instrument is not warmed up.

## SYSTem:TIMing:DATA

Set or query the stacking timing test data for the TTR500 instrument.

**Group** System commands

**Syntax**

```
SYSTem:TIMing:DATA <value>
SYSTem:TIMing:DATA?
```

**Arguments** `<value>::=<NR1>` is the stacking timing test data for the TTR500 instrument.

**Returns** `<NR1>`

**Examples** `SYST:TIM:DATA?` returns the stacking timing test data for the TTR500 instrument.

## SYSTem:UPReset:FILEname (No Query Form)

Set the user preset mode for the TTR500 instrument using a file.

**Group** System commands

**Syntax**

```
SYSTem:UPReset:FILEname <value>
```

**Arguments** `<value>::=<string>` is the preset file used to preset the TTR500 instrument.

**Examples** `SYST:UPR:FILE "user1"` sets the TTR500 to *user1* as the file for preset conditions.

## TRIGger:AVERage

Set or query the averaging trigger function.

**Group** Trigger commands

**Syntax**

```
TRIGger:AVERage <value>
TRIGger:AVERage?
```

**Arguments**

`<value>::=<boolean>`

**Returns**

1 means the averaging trigger is tuned on.

0 means the averaging trigger is turned off.

**Examples**

`TRIG:AVER 1` enables the averaging trigger.

`TRIG:AVER?` may return 0, which means the averaging trigger is disabled.

## TRIGger:EXTernal:DELay

Set or query the external trigger delay. This is the time taken by the VNA to start a measurement upon receipt of trigger when the source is external.

**Group**

Trigger commands

**Syntax**

```
TRIGger:EXTernal:DELay <value>
TRIGger:EXTernal:DELay?
```

**Arguments**

`<value>::=<NRf>` is the external trigger delay.

Range: 0 to 1 s

**Returns**

`<NRf>`

**Examples**

`TRIG:EXT:DEL 0.05` sets the external trigger delay to 0.05 s.

`TRIG:EXT:DEL?` may return 0.02, which means 0.02 s is the value set for the external trigger delay.

## TRIGger:[SEQuence]:[IMMediate] (No Query Form)

Generate a trigger sequence immediately, regardless of the SCPI manual trigger settings. This command generates a trigger and executes a measurement, independently of the SCPI manual trigger mode settings. Unlike `TRIGger:SEQuence:SINGle`, the execution of the object is complete at the same time as the trigger.

***NOTE.*** *An error occurs, if you execute this object while the trigger system is not in trigger wait mode and armed state.*

**Conditions** Use triggers to synchronize measurements with external events. The trigger function requires VectorVu-PC to have an active connection to a TTR500 unit. The trigger function does not work in simulator mode.

**Group** Trigger commands

**Syntax**

```
TRIGger:[SEQuence]:[IMMediate] <string>
```

**Arguments** `<string>` is the trigger sequence.

**Examples** `TRIG:SEQ:IMM` generates a trigger sequence immediately and completes a measurement.

## TRIGger:[SEQuence]:POINt

Set or query the point trigger function.

**Conditions** Use triggers to synchronize measurements with external events. The trigger function requires VectorVu-PC to have an active connection to a TTR500 unit. The trigger function does not work in simulator mode.

**Group** Trigger commands

**Syntax**

```
TRIGger:[SEQuence]:POINt <value>
TRIGger:[SEQuence]:POINt?
```

**Arguments**

```
<value>::=<boolean>
```

**Returns** 1 means the point trigger is turned on.

0 means the point trigger is turned off.

**Examples** `TRIG:SEQ:POIN 1` enables the point trigger.

`TRIG:SEQ:POIN?` may return 0, which means the point trigger is disabled.

## TRIGger:[SEQuence]:SCOPe

Set or query the scope for the trigger signal.

**Conditions** Use triggers to synchronize measurements with external events. The trigger function requires VectorVu-PC to have an active connection to a TTR500 unit. The trigger function does not work in simulator mode.

**Group** Trigger commands

**Syntax**

```
TRIGger:[SEQuence]:SCOPe { ALL | ACTive }
```

**Arguments** `ALL` sets trigger scope for all channels.

`ACTive` sets trigger scope only for the active channel.

**Returns** `ALL` means trigger scope is set to all channels.

`ACTive` means trigger scope is only set for the active channel.

**Examples** `TRIG:SEQ:SCOP ALL` sets trigger scope to all channels.

`TRIG:SEQ:SCOP?` may return ACTive, which means trigger scope is only set for the active channel.

## TRIGger:[SEQuence]:SINGle (No Query Form)

Generate a trigger immediately and execute a measurement, independently of the SCPI manual trigger mode settings. Unlike `TRIGger:SEQuence:IMMediate`, the execution of the command is complete when the measurement (of all sweep) initiated with this command is complete. You can wait for the end of the measurement using the OPC object.

***NOTE.*** *An error occurs, if you execute this command while the trigger system is not in trigger wait mode and armed state.*

**Conditions** Use triggers to synchronize measurements with external events. The trigger function requires VectorVu-PC to have an active connection to a TTR500 unit. The trigger function does not work in simulator mode.

**Group** Trigger commands

**Syntax**

```
TRIGger:[SEQuence]:SINGle <string>
```

**Arguments**

`<string>` is the trigger sequence.

**Examples**

`TRIG:SEQ:SING` generates a trigger sequence immediately and completes a measurement.

## TRIGger:[SEQuence]:SOURce

Set or query the source for the trigger.

**Conditions**

Use triggers to synchronize measurements with external events. The trigger function requires VectorVu-PC to have an active connection to a TTR500 unit. The trigger function does not work in simulator mode.

**Group**

Trigger commands

**Syntax**

```
TRIGger:[SEQuence]:SOURce { MANual | BUS | INTernal |
EXTernal }
TRIGger:[SEQuence]:SOURce?
```

**Arguments**

| Argument | Description |
|---|---|
| MANual | Indicates that the trigger source is manual. |
| BUS | Indicates that the trigger source is to SCPI commands. |
| INTernal | Indicates that the trigger source is internal. |
| EXTernal | Indicates that the trigger source is external. |

**Returns**

See Arguments.

**Examples**

`TRIG:SEQ:SOUR BUS` sets SCPI commands as the trigger source used to synchronize measurements with external events.

`TRIG:SEQ:SOUR?` may return MAN, which is the trigger source used to synchronize measurements with external events.

## *WAI (No Query Form)

Prevents the analyzer from executing further commands or queries until all pending operations are complete. This command allows you to synchronize the analyzer operations with your application program. For more details, refer to *Synchronizing Execution* (See page 3-9.).

**Group** IEEE common commands

**Syntax** `*WAI`

**Related Commands** *OPC

**Arguments** None

# Status and Events

# Status and Events

The SCPI interface in the VNA includes a status and event reporting system that allows you to monitor crucial events occurring in the instrument. The analyzer is equipped with 4 registers and 1 queue that conform to IEEE Std 488.2-1987. This section discusses these registers and queues along with status and event processing.

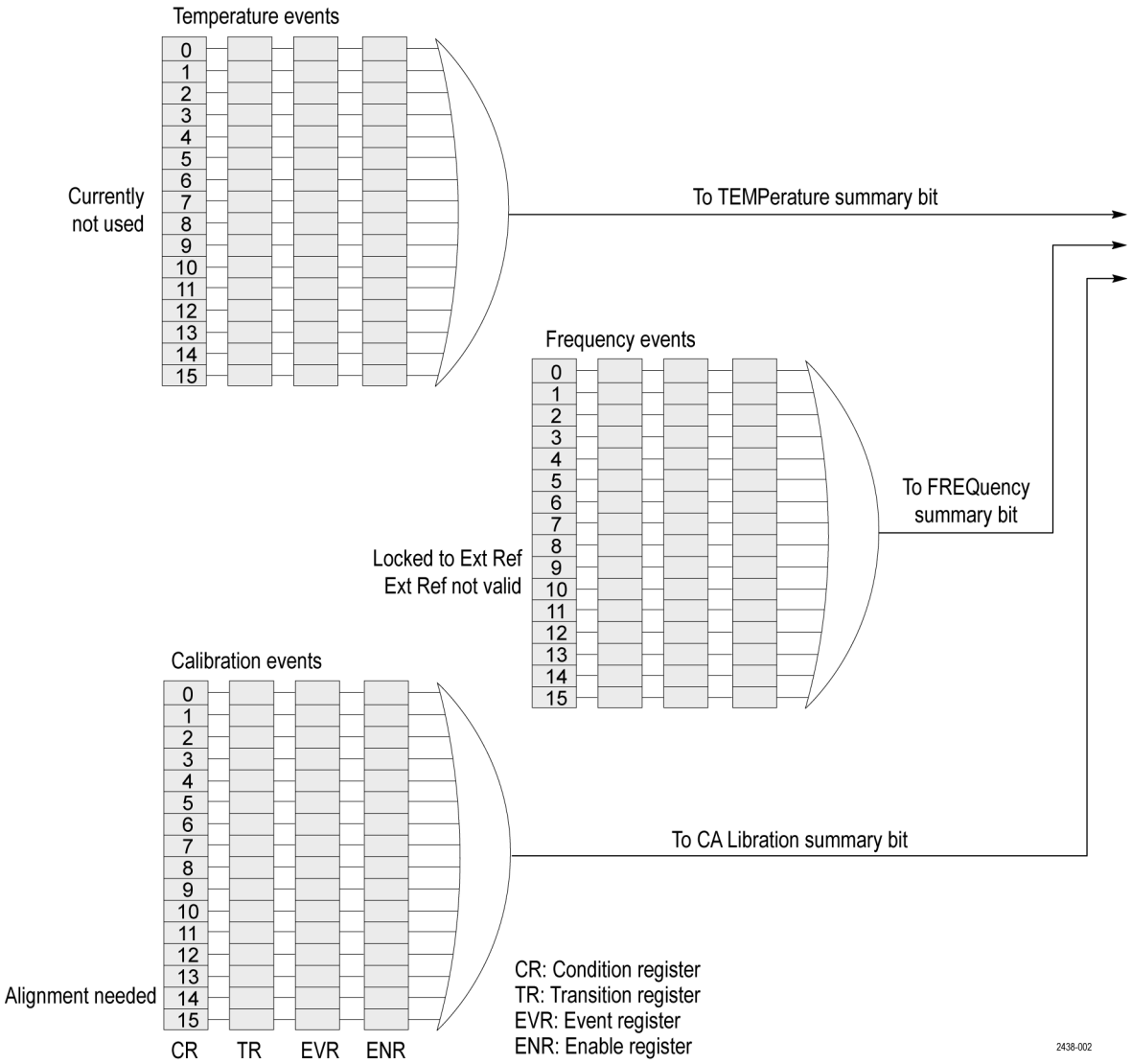
## Status and Event Reporting System

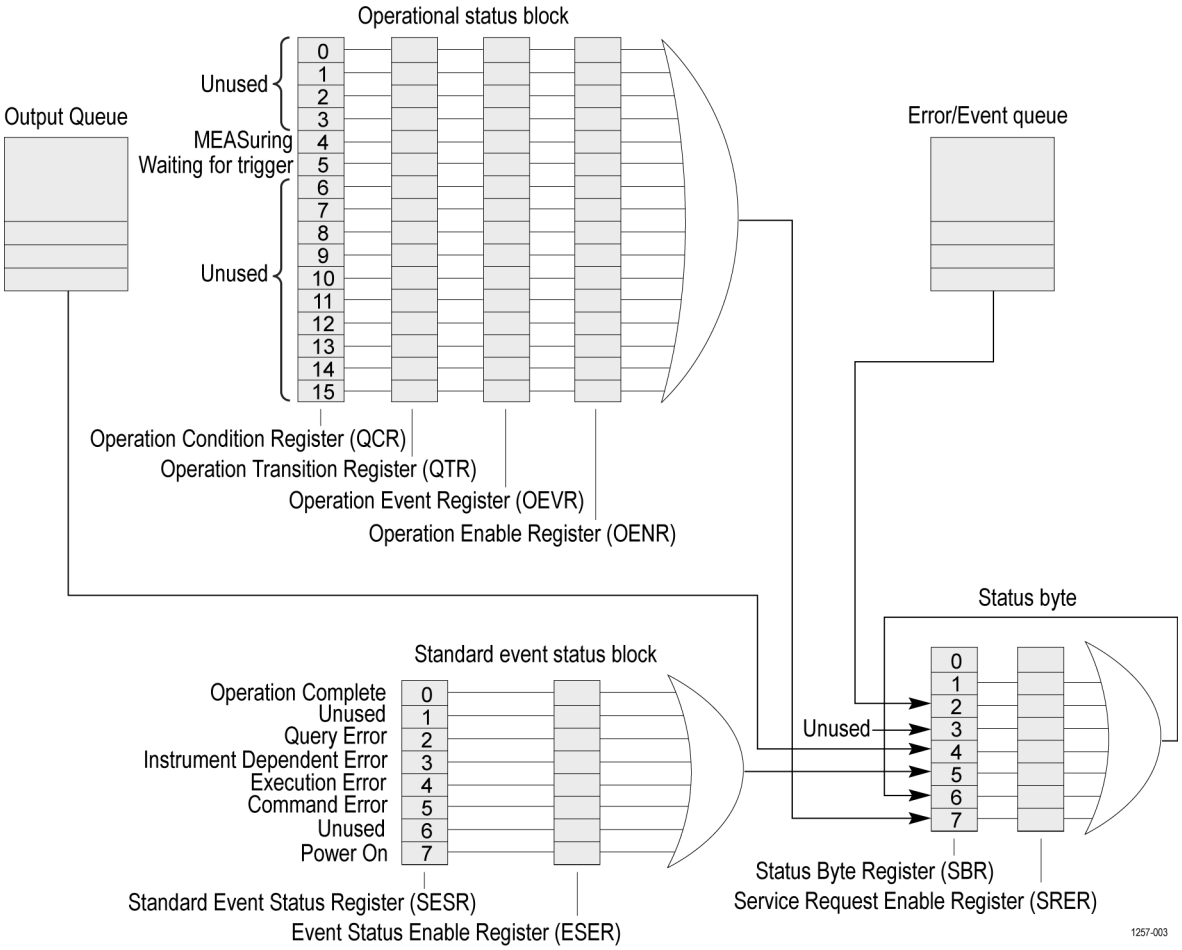
The status and event reporting mechanism in the TTR500 series analyzers has two major blocks:

- Standard Event Status
- Operation Status

The processes performed in these blocks are summarized in the Status Byte. The 2 blocks contain four types of registers:

| Register types | Description |
|---|---|
| Condition register | Records event occurrence in the instrument. Read only. |
| Transition register (pos/neg) | A positive transition filter allows an event to be reported when a condition changes from false to true.<br>A negative filter allows an event to be reported when a condition changes from true to false.<br>Setting both positive and negative filters true allows an event to be reported anytime the condition changes.<br>Clearing both filters disables event reporting. |
| Event register | Records events filtered by the transition register. Read only. |
| Enable register | Masks the event register to report in the summary bit. User-definable. |


Temperature events
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Currently not used
To TEMPerature summary bit
Frequency events
Locked to Ext Ref
Ext Ref not valid
To FREQuency summary bit
Calibration events
Alignment needed
To CA Libration summary bit
CR
TR
EVR
ENR
CR: Condition register
TR: Transition register
EVR: Event register
ENR: Enable register
2438-002




## Status Byte

The Status Byte contains these registers

- Status Byte Register (SBR)
- Service Request Enable Register (SRER)

### Status Byte Register (SBR)

The SBR is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2-1987. These bits are used to monitor the output queue, SESR and service requests, respectively. Contents of this register are returned when the *STB? query is used.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| OSS | RQS | ESB | MAV | QSS | EAV | — | — |

1257-004

| Bit | Description |
|---|---|
| 7 | Operation Summary Status (OSS). Summary of the operation status register. |
| 6 | Request Service (RQS)/Master Status Summary (MSS). When the instrument is accessed using the GPIB serial poll command, this bit is called the Request Service (RQS) bit and indicates to the controller that a service request has occurred (in other words, that the GPIB bus SRQ line is LOW). The RQS bit is cleared when serial poll ends. |
| 5 | Event Status Bit (ESB). This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed. |
| 4 | Message Available Bit (MAV). This bit indicates that a message has been placed in the output queue and can be retrieved. |
| 3 | Unused |
| 2 | Event Quantity Available (EAV). Summary of the Error Event Queue. |
| 1-0 | Unused |

### Service Request Enable Register (SRER)

SRER is made up of bits defined exactly the same as bits 0 to 7 in the SBR as shown in the following figure:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| OSS | — | ESB | MAV | — | — | — | — |

1257-005

Use `*SRE` to set the bits of the SRER.

Use `*SRE?` to read the contents of the SRER. Bit 6 must normally be set to 0.

You may use SRER to determine which events generate service requests.

The SRER bit 6 cannot be set.

The RQS is not maskable.

Generating a service request with the GPIB interface involves changing the SRQ line to LOW and making a service request to the controller. The result is that a status byte for which an RQS has been set is returned in response to serial polling by the controller

Use `*SRE` to set the bits of the SRER.

Use `*SRE?` to read the contents of the SRER. Bit 6 must normally be set to 0.

## Standard Event Status Block

This block reports the power on/off state, command errors, and the running state. It consists of the following 2 registers:

- Standard Event Status Register (SESR) — To read contents of SESR, use `*ESR?`.
- Event Status Enable Register (ESER) — To access ESER, use `*ESE`.

These registers are made up of the bits defined in the following figure and table:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PON | — | CME | EXE | IDE | QYE | — | OPC |

1257-006

| Bit | Description |
|---|---|
| 7 | Power On (PON). Indicates that the power to the instrument is on. |
| 6 | Not used |
| 5 | Command Error (CME). Indicates that a command error has occurred while parsing by the command parser was in progress. |
| 4 | Execution Error (EXE). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons:<br>■ When a value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument.<br>■ When the command could not be executed properly because the conditions for execution differed from those essentially required. |
| 3 | Instrument-Dependent Error (IDE). An instrument error has been detected. |
| 2 | Query Error (QYE). Indicates that a query error has been detected by the output queue controller. Query errors occur for one of the following reasons:<br>■ An attempt was made to retrieve messages from the output queue, despite the fact that the output queue is empty or in pending status.<br>■ The output queue messages have been cleared despite the fact that they have not been retrieved. |
| 1 | Not used. |
| 0 | Operation Complete (OPC). This bit is set with the results obtained from executing `*OPC`. It indicates that all pending operations have been completed. |

When an event occurs, the SESR bit corresponding to the event is set, resulting in the event being stacked in the error/event queue.

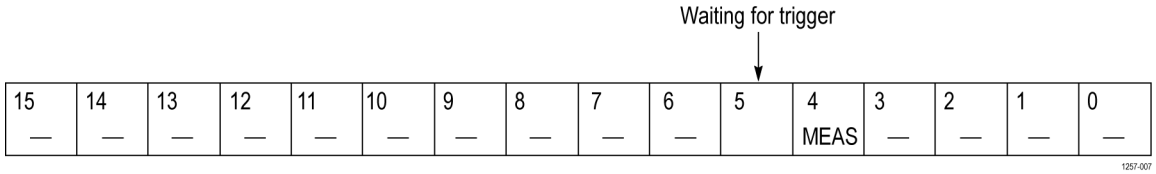
The SBR OAV bit is also set. If the bit corresponding to the event has also been set in the ESER, the SBR ESB bit is also set. When a message is sent to the Output Queue, the SBR MAV bit is set.

## Operation Status Block

The operation status block contains conditions that are part of the normal operation of the instrument. It consists of the following registers:

- Operation Condition Register (OCR)
- Operation Positive/Negative Transition Register (OPTR/ONTR)
- Operation Event Register (OEVR)
- Operation Enable Register (OENR)

These registers are made up of the bits defined in the following table and figure. Use `STATus:OPERation` to access the operation status register set.




| Bit | Description |
|---|---|
| 15 | Always zero (0) |
| 14 - 6 | Unused |
| 5 | Waiting for trigger. Indicates the TTR500 unit is stopped until a trigger is received. |
| 4 | Measuring (MEAS). Indicates the bit is set during measurement. The bit is reset after measurement.<br>"In measurement" means that one of the following commands is in execution:<br>■ INITiate commands<br>■ READ commands |
| 3 - 0 | Unused |

When the specified state changes in the OCR, its bit is set or reset. This change is filtered with a transition register, and the corresponding bit of the OEVR is set. If the bit corresponding to the event is also set in the OENR, the SBR OSS bit will also be set.

## Queues

There are two types of queues in the status reporting system used in the analyzer: output queues and event queues.

### Output Queue

The output queue is a FIFO (first in, first out) queue and holds response messages to queries, where they await retrieval. When there are messages in the queue, the SBR MAV bit is set.

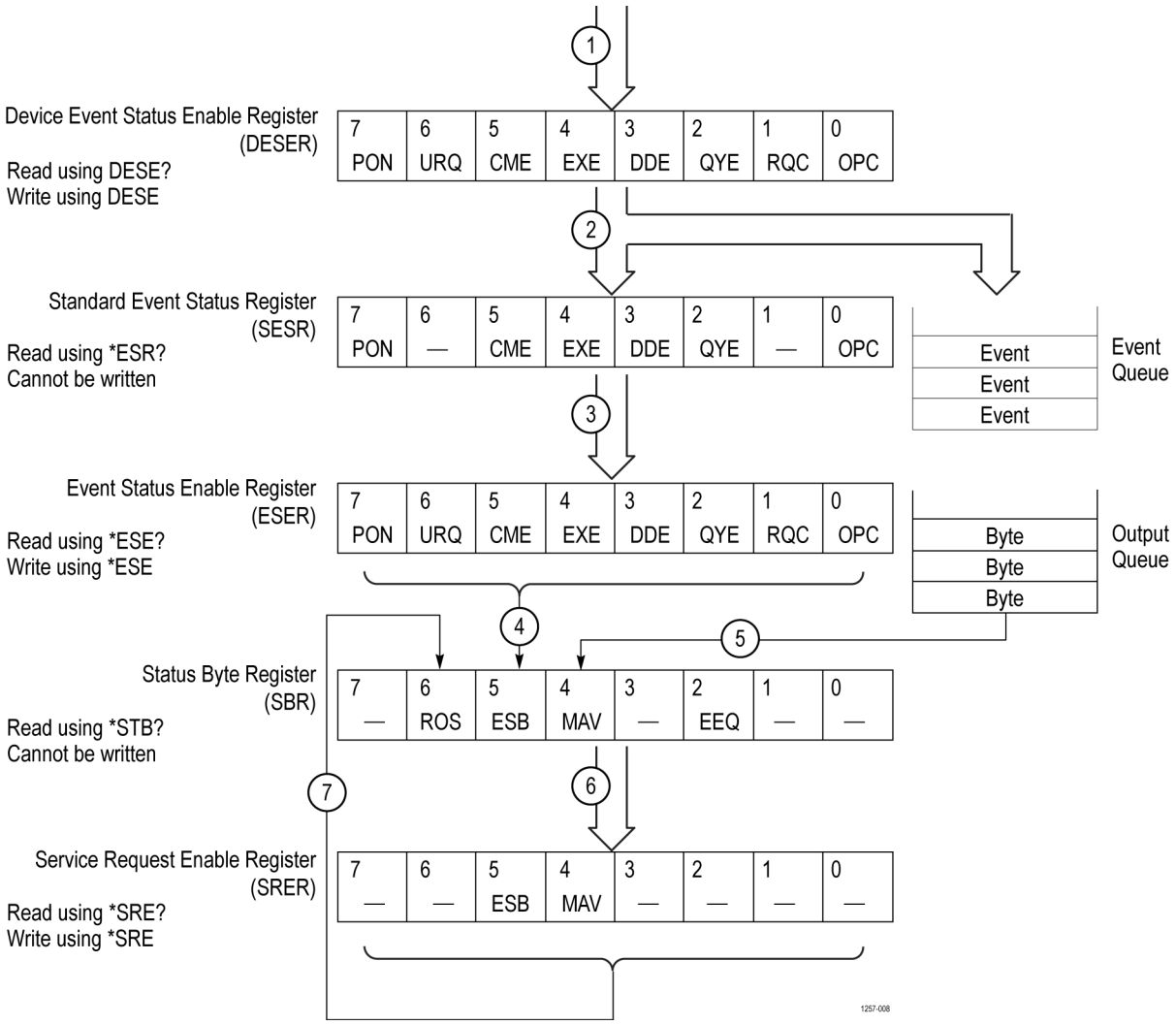
The output queue will be emptied each time a command or query is received, so the controller must read the output queue before the next command or query is issued. If this is not done, an error will occur and the output queue will be emptied; however, the operation will proceed even if an error occurs.

### Event Queue

The event queue is a FIFO queue and stores events as they occur in the analyzer. If more than 32 events occur, event 32 will be replaced with event code -350 ("Queue Overflow"). The error code and text are retrieved using the `SYSTem:ERRor` queries.

## Status and Event Processing Sequence

The following figure shows an outline of the sequence for status and event processing:




Below is the sequence for processing status and events:

1. If an event has occurred, the SESR bit corresponding to that event is set and the event is placed in the event queue.
2. A bit corresponding to that event in the ESER is set.
3. The SBR ESB bit is set to reflect the status of the ESER.
4. When a message is sent to the output queue, the SBR MAV bit is set.
5. Setting either the ESB or MAV bits in the SBR sets the respective bit in the SRER.
6. When the SRER bit is set, the SBR MSS bit is set and a service request is generated when using the GPIB interface.

## Synchronizing Execution

Almost all commands are executed in the order in which they are sent from the controller. The execution of each command is completed in a short period of time. However, `INITiate` commands perform data analysis in a different thread. Therefore, you can execute another command concurrently with an `INITiate` command.

These commands are designed so that the next command to be sent is executed without waiting for the previous command to be completed. In some cases, a process executed by another command must first be completed before these commands can be executed. In other cases, these commands must be completed before the next command is executed.

Use these options for command synchronization:

- Status and event reporting function
- Synchronizing commands

### Status and Event Reporting Function

In the following example, a `CALCulate` command is used to obtain measurement results while the Operation Condition Register (OCR) is used to provide synchronization:

```
*RST
    // Perform a system reset
:INITiate1:CONTinuous OFF;
    // Disable continuous initiation mode
:TRIGger:SOURce BUS;
    // Set trigger source to SCPI
:SYSTem:ERRor?
    // Query if an error has occurred
:INITiate1;
    // Initiate channel 1
*OPC?
    // Wait for channel 1 to initiate
:SYSTem:ERRor?
    // Query if an error has occurred
:TRIGger:SINGle;
    // Generate a trigger signal
*OPC?
    // Wait for measurement to complete
:SYSTem:ERRor?
    //Query if an error has occurred
:CALCulate1:DATA:SDATa?
    // Query the measurement results
```

The command waits for generation of SRQ.

## Synchronizing Commands

The IEEE-488.2 common commands include the following synchronizing commands:

- *OPC
- *OPC?
- *WAI

**Using the *OPC command.** `*OPC` sets the SESR OPC bit when all the operations for which it is waiting are completed. You can synchronize using `*OPR`. To check the status of the OPC bit, use `*ESR?`.

**Using the *OPC? query.** `*OPC?` writes ASCII code "1" into the output queue when all operations for which it is waiting are completed. You can provide synchronization using the command string as the following example:

```
ABOR:INIT:IMM:*OPC?
```

The command waits until "1" is written into the output queue. When the output queue is read, in the event that the system time-out is shorter than the time required to execute the command, a time-out will occur before the data is written into the queue.

**Using the *WAI Command.** Use `*WAI` for command synchronization, as shown in the following example:

```
ABOR:INIT:IMM:*WAI
    // Wait for the *WAI process to provide synchronization
```

Commands and queries subsequent to *WAI are held off until the wait completes.

# Error messages

## IEEE error messages

| Error ID | Message | Description |
|---|---|---|
| 100 | Command syntax error | SCPI command error |
| 101 | Invalid character | Command contains invalid character |
| 103 | Invalid separator | Delimiter expected, different character sent |
| 104 | Data type error | Wrong data element type received. |
| 105 | Parameter not allowed | Number of parameters exceeds command syntax. |
| 106 | Missing parameter | Number of parameters is less than command syntax. |
| 107 | Command not supported | Command has valid SCPI syntax but not supported by instrument. |
| 108 | Header suffix out of range | Unit of the header is out of range. |
| 110 | Numeric data error | Numeric parameter has invalid syntax. |
| 111 | Invalid character in number | Character is invalid in numeric parameter - e.g. letter found in decimal value. |
| 112 | Exponent too large | Absolute value of exponent exceeds 32,000. |
| 113 | Too many digits. | More than 255 digits in number |
| 114 | Numeric data not allowed | Numeric data element received at invalid position. |
| 120 | Invalid suffix | Suffix has invalid syntax or value. |
| 121 | Suffix too long | Unit has more than 11 characters. |
| 122 | Suffix not allowed | Suffix is attached to numeric value which does not have units. |
| 130 | Invalid character data | Invalid character found in character data element. |
| 131 | Character data not allowed | Character data element was received at invalid position. |
| 140 | String data error | Content of the string has syntax error. |
| 141 | Invalid string data | Character string is invalid. |
| 142 | String data not allowed | A character string data element was received at invalid position. |
| 143 | String too long | String longer than 255 characters |
| 150 | Invalid block data | Block data is invalid, possibly due to block data length mismatch. |
| 160 | Expression error | Expression syntax error |
| 161 | Invalid expression | Expression data element is invalid. |
| 162 | Expression data not allowed | Expression data element was received at invalid position. |
| 200 | Execution error | Error in execution. |
| 201 | Measurement in progress - init ignored | Init command received while measurement is in progress. Init command is ignored. |

| | | |
|---|---|---|
| 210 | Parameter error | General parameter error. |
| 211 | Unexpected trigger - ignored | Trigger is detected when not in "waiting for trigger" state. |
| 212 | Data out of range | Data element out of range was received. |
| 214 | Illegal parameter value | Measurement parameter is invalid - e.g. source/receive port are invalid. |
| 220 | File not found | Specified file not found. |
| 221 | File name error | File name syntax error |
| 301 | Error queue overflow | Error occurred when there is only one slot available in error queue. |
| 401 | Query interrupted | Data bytes received before response of previous query. |
| 402 | Query not terminated | Instrument designated as talker and incomplete program message is received. |
| 403 | Query deadlocked | Both input and output buffers are full. |
| 404 | Query not terminated after indefinite response | Query asking for indefinite response is followed by new query. |
| 500 | Data array contains NAN values. | Data array sent to user contains NAN values. |

## VNA error messages

This is a list of instrument-specific error messages that appear in the Instrument Status bar to indicate an improper setting or incorrect action:

| Error ID | Message | Description |
|---|---|---|
| 10 | Unable to calculate calibration data. Missing standards. | Some standards were not measured for selected calibration method. |
| 11 | Duplicate port numbers | Identical port numbers specified in a list of ports. |
| 12 | No calibration method selected. | Unable to save calibration data before selecting a calibration method. |
| 13 | Specified error term is invalid for the selected calibration method. | This error term is not applicable for the selected calibration method. |
| 14 | Normal calibration not allowed in frequency offset mode | Unable to execute normal calibration frequency offset mode. |
| 15 | Scalar mixer calibration not allowed when frequency offset mode is disabled. | Scalar mixer calibration not allowed when frequency offset mode is disabled. |
| 16 | Unable to perform partial override. No calibration method specified. | Unable to perform partial override when the calibration method is not specified. |
| 17 | Correction not enabled. No calibration data exists. | Error correction cannot be enabled because no calibration data exists. |

| Error ID | Message | Description |
|---|---|---|
| 20 | eCal module does not support specified number of ports. | Unable to run multi-port calibration. eCal module has insufficient number of ports. |
| 21 | eCal failure | Unable to configure or read from eCal module. |
| 22 | eCal not connected in RF path | eCal auto-detect does not recognize RF connectivity between the VNA and the eCal module |
| 23 | Unable to perform confidence check for mixed mode S-parameters. | Unable to perform confidence check for mixed mode S-parameters. |
| 24 | Characterization not found in eCal module. | Characterization entry does not exist in eCal module memory. |
| 30 | Target value not found | Target value not found within specified excursion in target search. |
| 31 | Peak not found. | Peak not found within specified excursion in peak search. |
| 40 | Selected channel not on display. | Selected active channel is not on display. |
| 41 | Selected trace does not exist. | Selected active trace does not exist. |
| 42 | No valid memory trace | Unable to perform memory operation where there is no valid memory trace. |
| 43 | Time domain processing not supported. | Unable to perform time domain transform or gating. |
| 44 | Fixture simulator not allowed in frequency offset mode. | Unable to run fixture simulator in frequency offset mode. |
| 45 | Auto port extension not allowed in power sweep or frequency offset mode. | Auto port extension not allowed in power sweep or frequency offset mode. |
| 46 | No valid measurement to save in file. | Unable to save trace data. No valid trace available. |
| 50 | Frequency out of range | Selected frequency outside instrument range. |
| 60 | Power meter not found | Unable to locate power meter based on specified VISA resource name. |
| 61 | Power meter not stable | Power meter reading exceeds tolerance or fails to stabilize within predefined time. |
| 62 | Signal generator not found | Unable to locate signal generator based on specified VISA resource name. |
| 63 | Unable to control external signal generator. | Unable to control external signal generator. |
| 64 | Unable to communicate with power meter. | Unable to communicate with power meter. |

| Error ID | Message | Description |
|---|---|---|
| 65 | Power meter already claimed. | Power meter already claimed by another application. |
| 66 | Selected power sensor does not support the configured sweep range. | Selected power sensor does not support the configured sweep range. Unable to perform power calibration. |
| 100 | Load failed. | Load from file failed. |
| 101 | Save failed. | Save to file failed. |
| 102 | THRU standard should be defined with S2P file. | THRU standard should be defined with S2P file. |
| 103 | Standard should be defined with S1P file. | Standard should be defined with S1P file. |
| 104 | Factory calibration not found. | Unable to locate factory calibration zip file for this instrument. Default calibration used instead. |
| 200 | Option not installed | Unable to perform optional function. This option is not installed in the instrument. |
| 210 | PLL not locked | Measurement data is not valid. At least one frequency point resulted in an unlocked PLL |
| 220 | Power trip event. RF turned off | Power received was too high. Source turned off to prevent damage to VNA. |
| 230 | Self test failure | Instrument self test failed. |
| 300 | Unable to estimate adapter length in zero span. | Unable to estimate adapter length in zero span. |
| 400 | Invalid equation expression | Invalid equation is specified in the equation editor |
| 401 | Invalid equation label | Invalid equation label. |
| 500 | Instrument disconnected | The instrument has been disconnected. |
| 501 | Instrument is not connected. | Unable to connect to instrument that is not connected to host. |
| 502 | Instrument connection failed. | Instrument connection failed. |
| 503A | Instrument connection failed. | Instrument already claimed. |
| 504 | Instrument connection failed. | Instrument not activated. |
| 505 | Instrument connection failed. | Device is not connected to USB. |
| 506A | USB Storage set state failure. | Unable to set state of USB Storage. |
| 507 | USB Storage get state failure. | Unable to get state of USB Storage. |
| 508 | Instrument activation failed. | Instrument already claimed. |
| 509 | Instrument activation failed. | Instrument already activated. |
| 510 | Instrument activation failed. | Device is not connected to USB. |

| Error ID | Message | Description |
|---|---|---|
| 511 | Instrument temperature has exceeded safe operating range. | The temperature of TTR500 unit exceeded safe operating range. VectorVu-PC has switched to offline mode. |
| 601 | Unable to load power meter DLL. | Unable to load power meter DLL. Verify DLL is installed and in execution path. |
| 602 | Incompatible power meter DLL. | Power meter installation is not compatible with this application. |
| 603 | Power meter initialization failed. | Power meter initialization failed. |
| 604 | Power meter measurement failed. | Power meter measurement failed. |
| 700 | Insufficient memory for current sweep settings. Unable to start sweep recording | Insufficient memory for current sweep settings. Unable to start sweep recording. |
| 1000 | Calibration interpolated. | Measurement and calibration stimulus points are not identical. Calibration data was interpolated. |
| 1001 | Measurement conditions not identical to calibration conditions. | Measurement and calibration conditions are not identical. Error correction can be inaccurate. |
| 1002 | Change in global power or IF BW. Segment table updated accordingly. | Global power or IF BW has changed. The per-segment power /IF BW were overridden in the segment table. |

## Warnings and notifications

This is a list of warning messages and notifications that display in the Instrument Status bar.

### Warnings

| Warning ID | Message | Description |
|---|---|---|
| 1000 | Calibration interpolated | Measurement and calibration stimulus points are not identical. Calibration data was interpolated |
| 1001 | Measurement conditions not identical to calibration | Measurement and calibration conditions (e.g. power, IF BW) are not identical. Error correction might not be accurate |
| 1002 | Changed global power or IF BW - updated segment table accordingly | Global power or IF BW changed - the per-segment power / IF BW were overridden in the segment table |

### Notifications

| Notification ID | Message | Description |
|---|---|---|
| 2000 | Instrument connected | A TTR500 unit has been connected. |

# Index

## Symbols and Numbers



## A



## C

## D

## E



## I



## M



## O



## R



## S

## T

## W