SCPI Programming Manual

S331L Site Master™

Handheld Cable and Antenna Analyzer

Anritsu Company
490 Jarvis Drive
Morgan Hill, CA 95037-2809
USA
http://www.anritsu.com

Part Number: 10580-00322
Revision: E
Published: May 2016

NOTICE

Anritsu Company has prepared this manual for use by Anritsu Company personnel and customers as a guide for the proper installation, operation and maintenance of Anritsu Company equipment and computer programs. The drawings, specifications, and information contained herein are the property of Anritsu Company, and any unauthorized use or disclosure of these drawings, specifications, and information is prohibited; they shall not be reproduced, copied, or used in whole or in part as the basis for manufacture or sale of the equipment or software programs without the prior written consent of Anritsu Company.

UPDATES

Updates, if any, can be downloaded from the Anritsu Website at:
http://www.anritsu.com

For the latest service and sales contact information in your area, please visit:
http://www.anritsu.com/contact-us

Table of Contents

Chapter 1 — General Information

1-1 About this Manual

This SCPI Programming Manual provides information for remote operation of the Site Master S331L, Cable and Antenna Analyzer, using commands sent from an external controller through the USB or Ethernet connection.

This Programming Manual includes the following:

- An overview of the USB and Ethernet connections to the instrument.
- An overview of Standard Commands for Programmable Instruments (SCPI) command structure and conventions.
- The IEEE common commands that are supported by the instruments.
- A complete listing and description of all the SCPI commands that can be used to remotely control functions of the instrument. The commands are organized by measurement mode starting in Chapter 3.

This manual is intended to be used in conjunction with the Site Master S331L User Guide. Refer to the instrument user guide for general information about the instrument, including equipment setup and operating instructions.

1-2 Introduction

This chapter provides a general description of remote programming setup and interface using USB or Ethernet, and sending SCPI commands to the instrument.

1-3 Remote Operation Setup and Interface

Remote operation of the instrument is accomplished via the USB or Ethernet interface. The following paragraphs provide information about the interface connections, cable requirements, and setting up remote operation.

USB Interface Connection and Setup

The Universal Serial Bus (USB) architecture is a high-performance networking standard that is considered “plug and play” compatible. The USB driver software is automatically detected and configured by the operating system of the devices that are connected to the bus. The instrument conforms to the USB 2.0 standard and is a USB “Hi-speed” device that supports data rates of up to 480 Mbps with the following restrictions:

- One USB network can support up to 127 devices
- The maximum length of USB cables between active devices is 5 meters (for USB 2.0) and 3 meters (for USB 1.0)

You must have NI-VISA 2.5 or later installed on the controller PC and must select the VISA library (visa32.dll) as a reference in a Visual Basic project. For remote USB control, the controlling PC needs to have a version of VISA installed that supports USBTMC (USB Test and Measurement Class) devices.

USB Interface, Type Mini-B

The USB 2.0 Mini-B device connector is used to connect the instrument directly to a PC. The first time the instrument is connected to a PC, the normal USB device detection by the computer operating system takes place.

1. Power on the instrument and controller PC and wait for the systems to power up completely.
2. Connect the USB cable Mini-B connector to the instrument.
3. Connect the USB cable A connector to the controller PC USB host port. The controller PC should indicate “New Hardware Found” if the combination of USB VID/PID/Serial Number has never been connected to this controller PC.

Figure 1-1. USB Found New Hardware Wizard

4. Select to allow the Wizard to search for and install the USB software automatically.

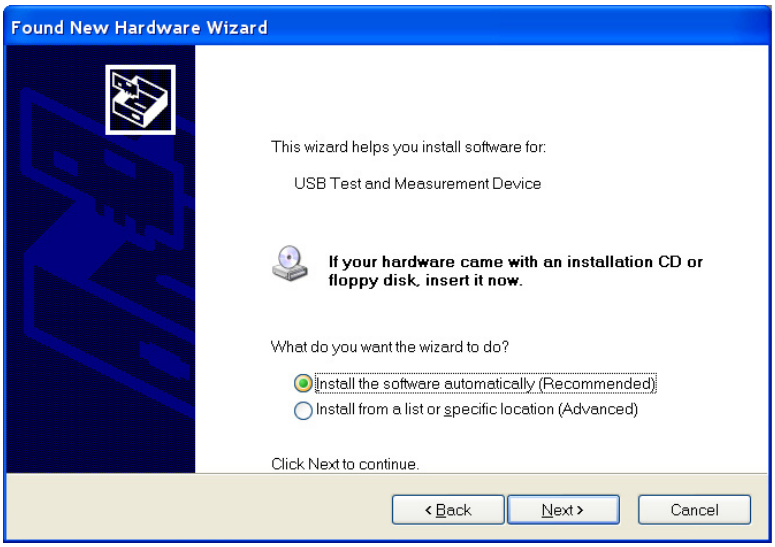


Figure 1-2. USB Found New Hardware Wizard

5. After the software installs, close the Wizard by clicking Finish.

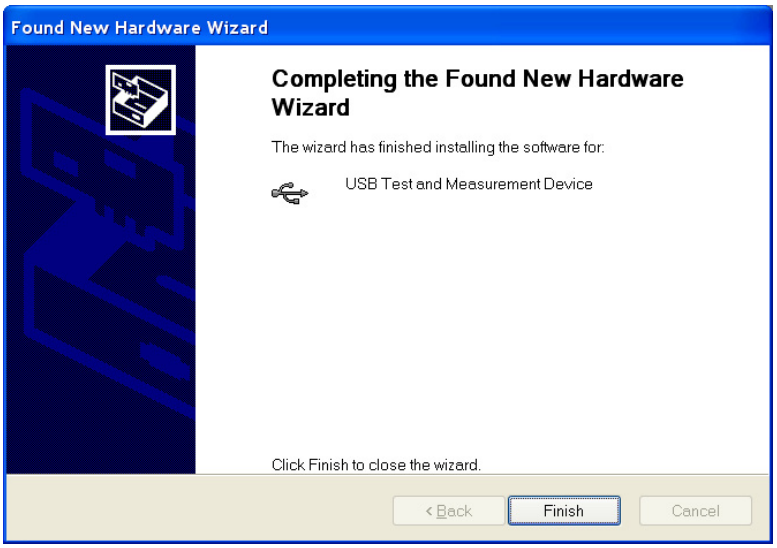


Figure 1-3. USB Found New Hardware Wizard

Ethernet Interface Connection and Setup

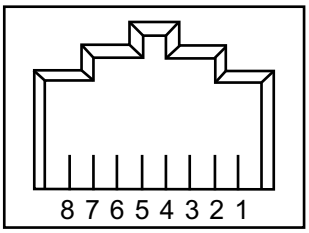
The Site Master S331L fully supports the IEEE-802.3 standard. Most Site Master functions (except power On/Off) can be controlled via an Ethernet connection to a PC that is connected directly (with an Ethernet cross-over cable) or through a network.

Ethernet networking uses a bus or star topology in which all of the interfacing devices are connected to a central cable called the bus, or are connected to a hub. Ethernet uses the CSMA/CD access method to handle simultaneous transmissions over the bus. CSMA/CD stands for Carrier Sense Multiple Access/Collision Detection. This standard enables network devices to detect simultaneous data channel usage (called a collision) and provides for a contention protocol. When a network device detects a collision, the CSMA/CD standard dictates that the data is retransmitted after waiting a random amount of time. If a second collision is detected, then the data are again retransmitted after waiting twice as long. This is known as exponential back off.

The TCP/IP setup requires the following:

- USB-to-Ethernet Dongle: The S331L requires the use of an external USB-to-Ethernet dongle, such as Anritsu part number 2000-1810-R, to enable TCP/IP communication.
- IP Address: Every computer/electronic device in a TCP/IP network requires an IP address. An IP address has four numbers (each between 0 and 255) separated by periods. For example: 128.111.122.42 is a valid IP address.
- Subnet Mask: The subnet mask distinguishes the portion of the IP address that is the network ID from the portion that is the station ID. The subnet mask 255.255.0.0, when applied to the IP address given above, would identify the network ID as 128.111 and the station ID as 122.42. All stations in the same local area network should have the same network ID, but different station IDs.
- Default Gateway: A TCP/IP network can have a gateway to communicate beyond the LAN that is identified by the network ID. A gateway is a computer or electronic device that is connected to two different networks and can move TCP/IP data from one network to the other. A single LAN that is not connected to other LANs requires a default gateway setting of 0.0.0.0. If you have a gateway, then the default gateway would be set to the appropriate value of your gateway.
- Ethernet Address: An Ethernet address (also known as a MAC address) is a unique 48-bit value that identifies a network interface card to the rest of the network. Every network card has a unique ethernet address permanently stored into its memory.

Table 1-1. 8-pin Ethernet RJ45 Connector Pinout Diagram



Pin	Name	Description	Wire Color
1	TX+	Transmit data (> +3 volts)	White/Orange
2	TX–	Transmit data (< –3 volts)	Orange
3	RX+	Receive data (> +3 volts)	White/Green
4	—	Not used (common mode termination)	Blue
5	—	Not used (common mode termination)	White/Blue
6	RX–	Receive data (< –3 volts)	Green
7	—	Not used (common mode termination)	White/Brown
8	—	Not used (common mode termination)	Brown

Connectivity

TCP/IP connectivity requires setting up the parameters that are described at the beginning of this section. The following is a brief overview of how to set up a general LAN connection on the Site Master.

Note	You may need to consult your network documentation or network administrator for assistance in configuring your network setup.

Site Master LAN Connections

The S331L requires the use of an external USB-Ethernet dongle, such as Anritsu part number 2000-1810-R, to connect the Site Master to a local area network (LAN). Integrated into this dongle are two LEDs (Light Emitting Diodes). The amber LED indicates the speed of the LAN connection (ON for 100 Mb/s and OFF for 10 Mb/s), and the green LED flashes to show that LAN traffic is present. The instrument IP address is set automatically by using Dynamic Host Configuration Protocol (DHCP). DHCP is an Internet protocol that automates the process of setting IP addresses for devices that use TCP/IP, and is the most common method of configuring a device for network use. After the Ethernet cable is connected to the instrument, go to System, Status, Connectivity Info to view the IP address that the instrument has been assigned.

1-4 Sending SCPI Commands

SCPI commands can be sent to the instrument through any Virtual Instrument Software Architecture (VISA) controller. VISA is a commonly used API in the Test and Measurement industry for communicating with instruments from a PC. The physical connection between the PC and the instrument is USB or Ethernet. NI-VISA is the National Instruments implementation of the VISA I/O standard. Information and downloads are available at: http://www.ni.com/visa/

The following example describes the verification that a VISA controller can interact with the instrument. The images shown and the instructions for your instrument and software may differ from the examples.

USB Connectivity

1. On the PC, run NI Measurement & Automation Explorer or VISA Interactive Control and double-click on the TMC Class instrument.

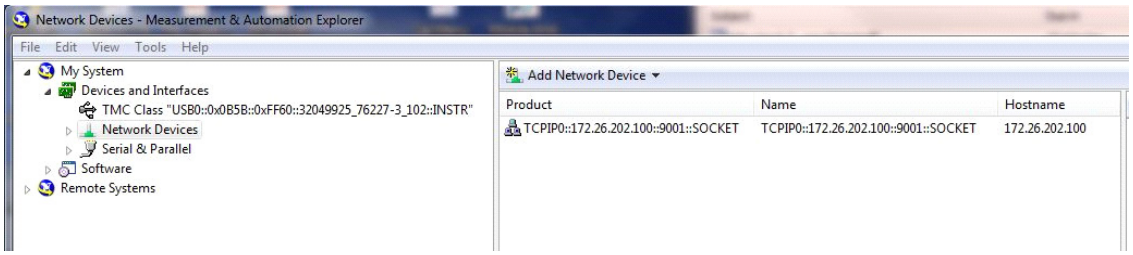


Figure 1-4. Figure 1-4. NI Measurement & Automation Explorer

2. Verify that the USB Settings list the correct Manufacturer, Model, and Serial Number, as shown in the example below.

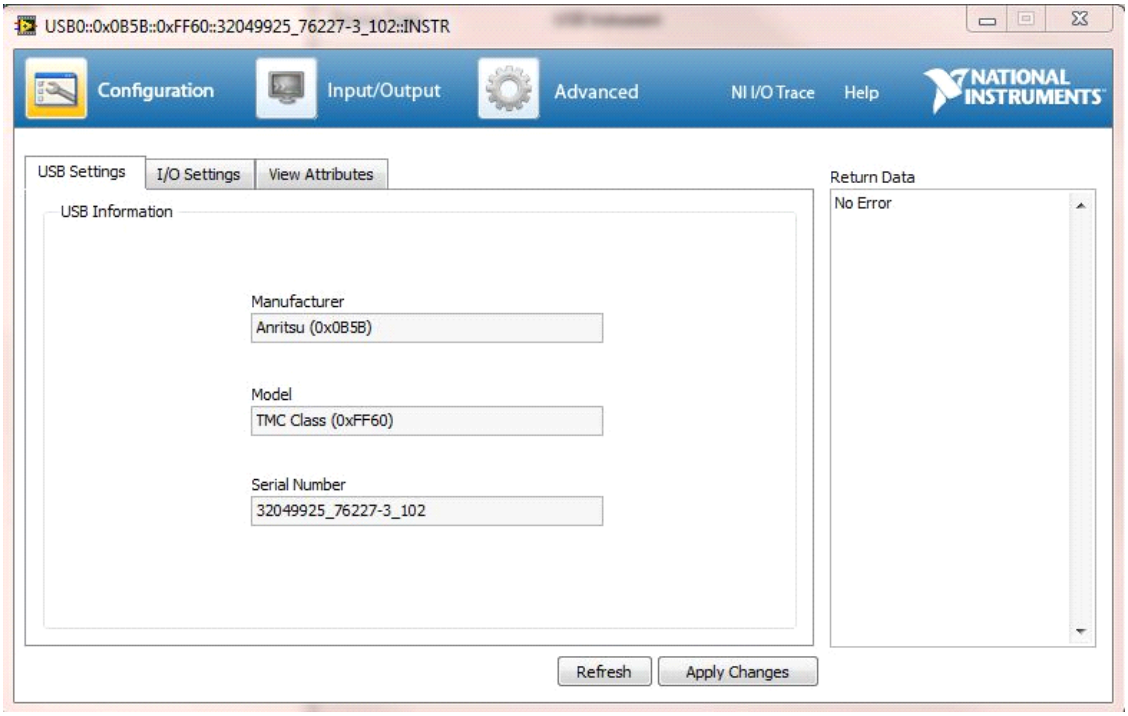


Figure 1-5. NI VISA Interactive Control USB Configurations / Settings Tab

3. Select the Input/Output Basic I/O tab and execute the default *IDN? Query. If the PC is connected to the instrument the command returns the following information from the Buffer: manufacturer name (“Anritsu”), model number/options, serial number, and firmware package number, as shown in the example below.

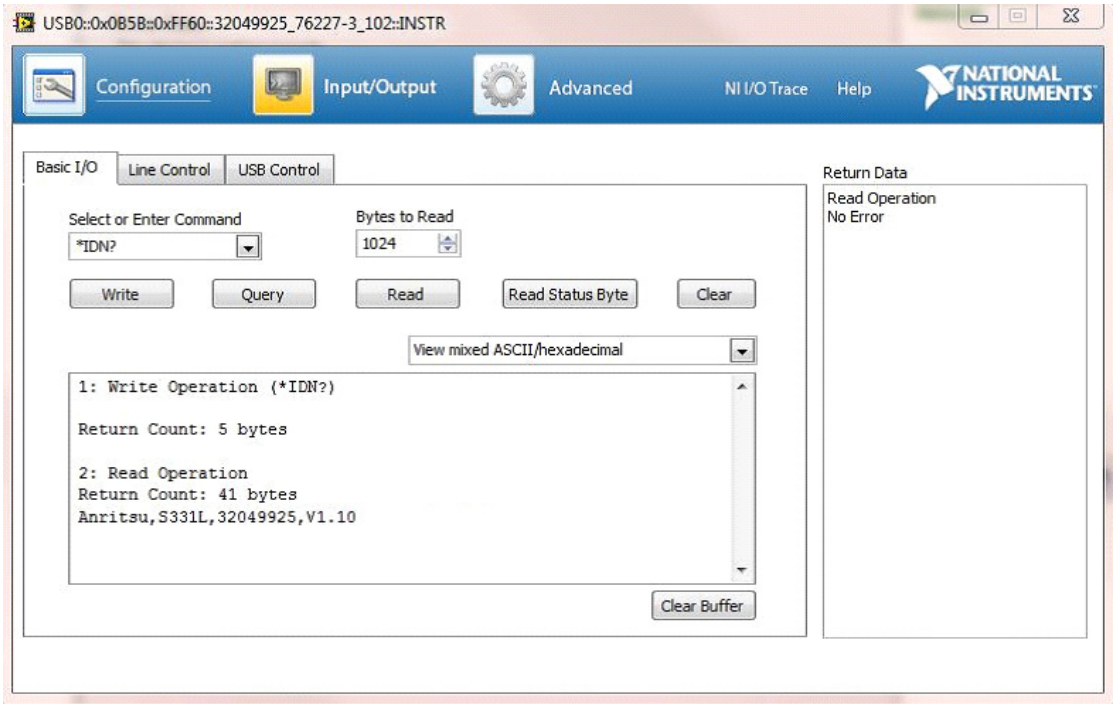


Figure 1-6. NI VISA Interactive Control USB Basic I/O Tab

Ethernet Connectivity

1. On the PC, run NI Measurement & Automation Explorer or VISA Interactive Control and create a new LAN Resource under Network Devices. Add the TCP/IP resource using a Manual Entry of Raw Socket, as shown in the example below.

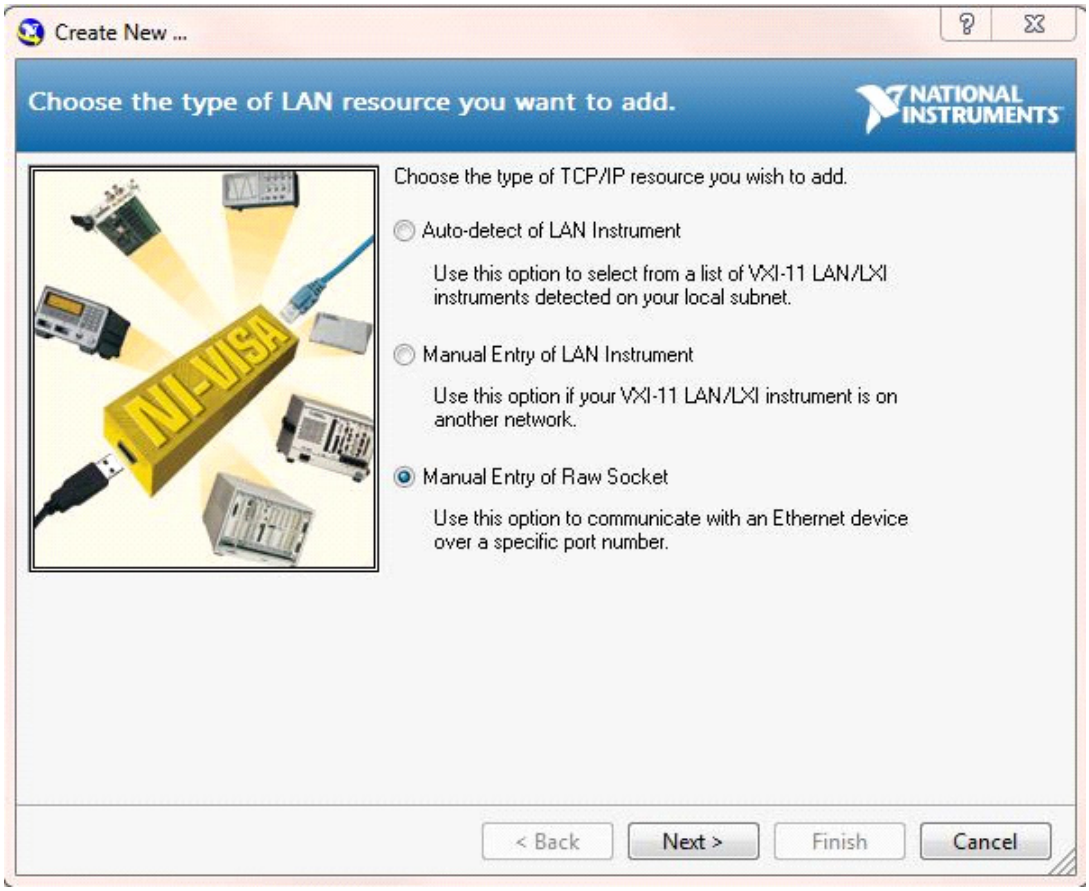


Figure 1-7. NI VISA Interactive Control LAN Resource Addition Using Raw Socket

2. Enter the IP address that the instrument has acquired (go to System, Status, Connectivity Info). Enter the port number as 9001, as shown in the example below.

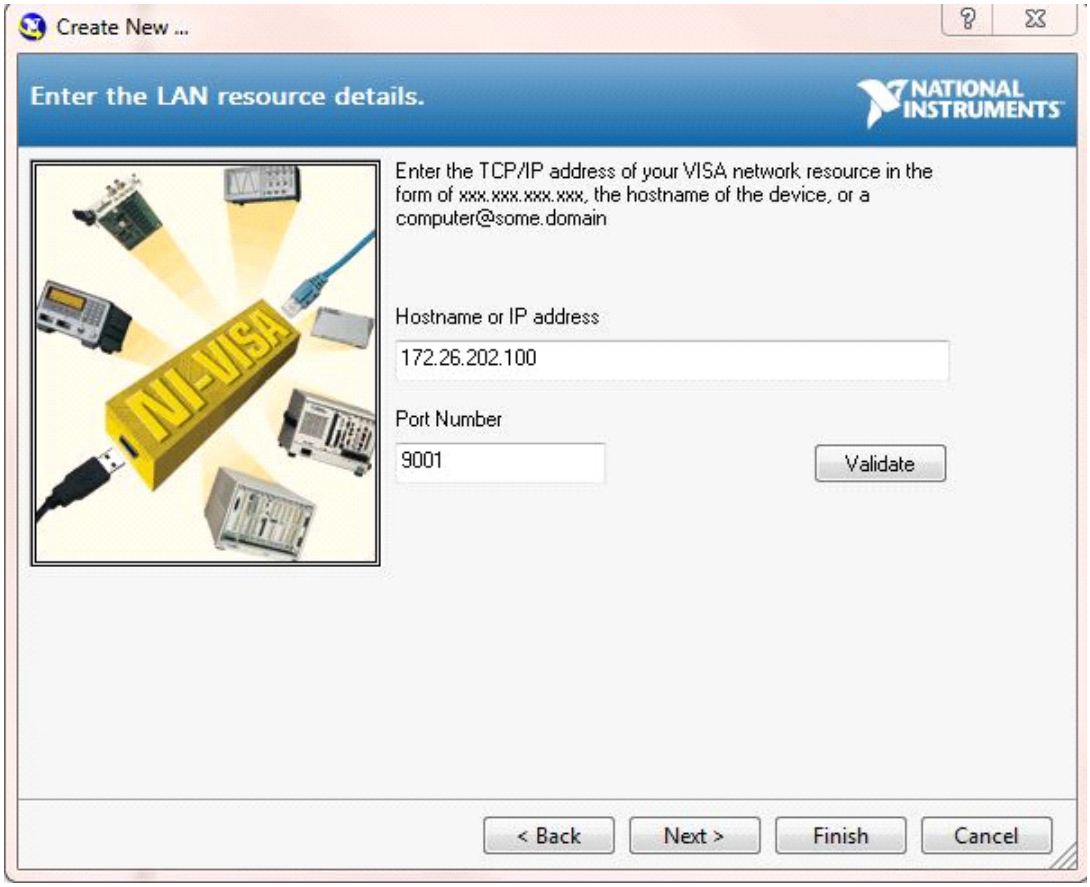


Figure 1-8. NI VISA Interactive Control LAN Resource Settings of IP Address and Port Number

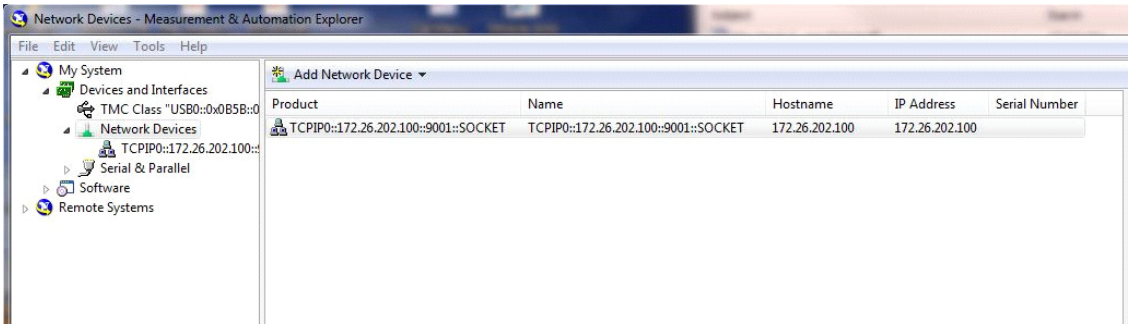


Figure 1-9. NI VISA Interactive Control LAN Resource Validated

3. Select the Configuration I/O settings tab and verify that the Termination Methods are set as shown in the example below.

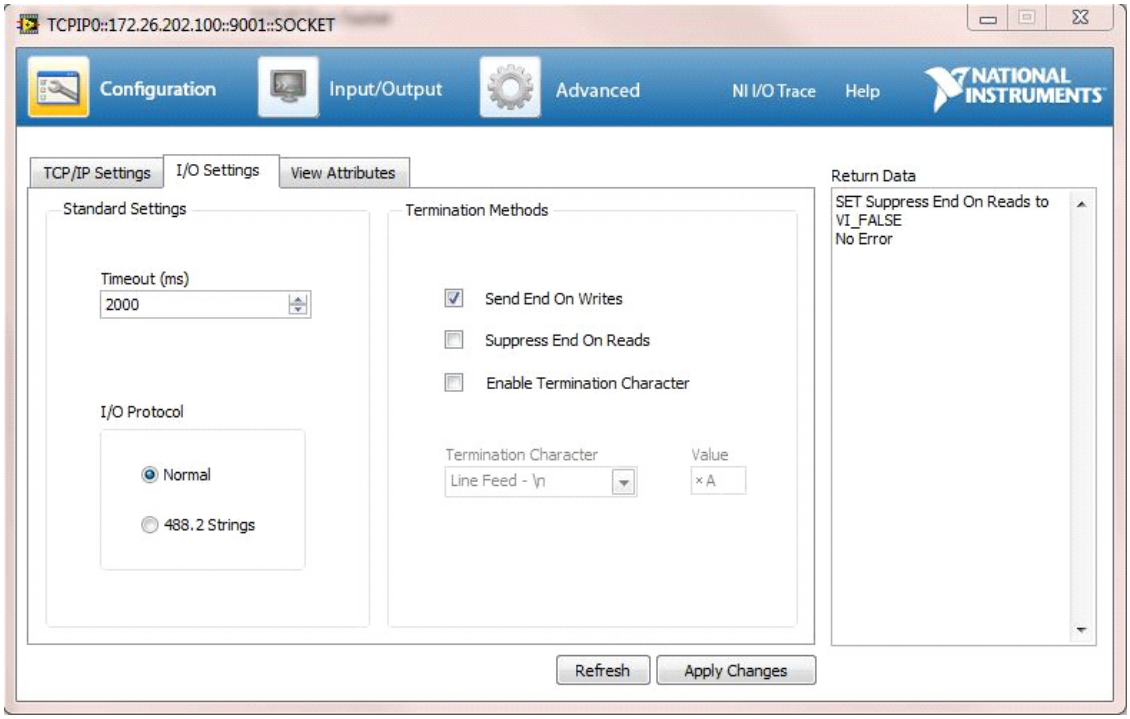


Figure 1-10. NI VISA Interactive Control LAN Resource I/O Termination Method Settings

4. Select the Input/Output Basic I/O tab and execute the default *IDN? Query. If the PC is connected to the instrument the command returns the following information from the Buffer: manufacturer name (“Anritsu”), model number/options, serial number, and firmware package number, as shown in the example below.

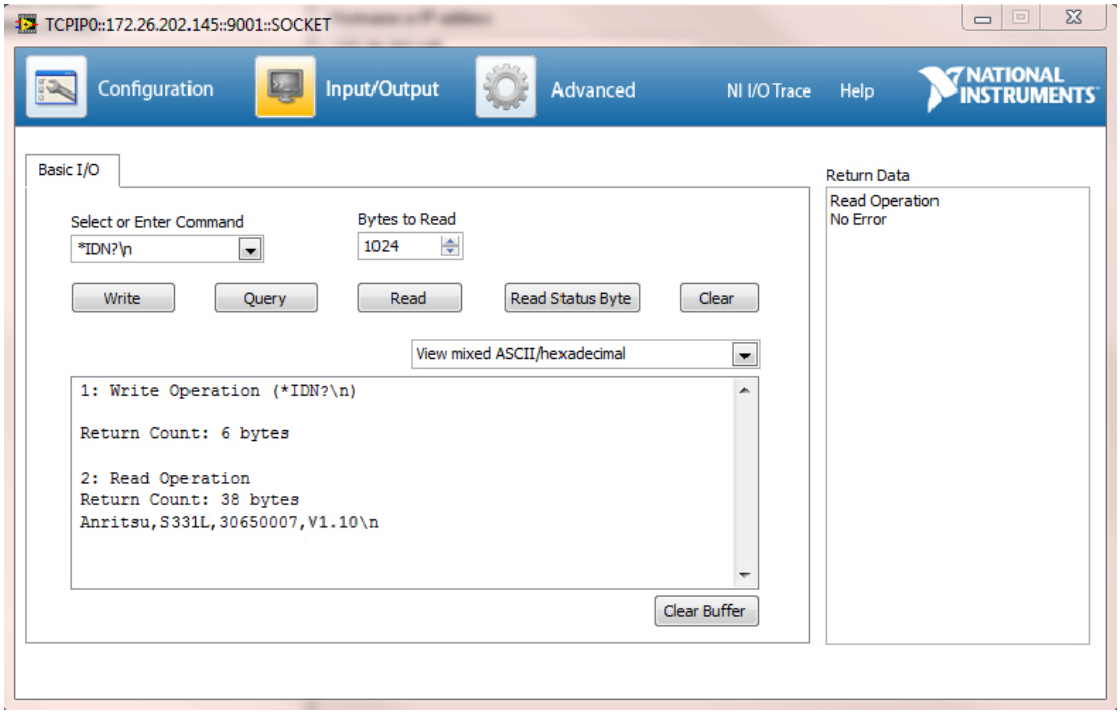


Figure 1-11. NI VISA Interactive Control USB Basic I/O Tab

Note	When sending SCPI commands over Ethernet, it is required to send a newline termination character at the end of each command. In the example above, a newline character ("\n" in this case, but could be different depending on your programming environment) was used to terminate the *IDN? command.

Note	When sending query commands over raw socket, the entire buffer must be read before the next query command is sent. Each query result is terminated by a newline to help identify the end of the query response. Query read operations could be broken into multiple reads, if necessary.

Note	When using raw socket connections, you must close a session before opening a new one or before switching to a new protocol (such as USB). If you try to open a new session or switch protocols without first closing the previously opened session, you may lose communications with the instrument and not be able to reconnect until you reboot the instrument.

Chapter 2 — Programming with SCPI

2-1 Introduction

This chapter provides an introduction to Standard Commands for Programming Instruments (SCPI) programming that includes descriptions of the command types, hierarchical command structure, command subsystems, data parameters, and notational conventions.

2-2 Introduction to SCPI Programming

Anritsu instruments can be operated with the use of SCPI commands. SCPI is intended to give the user a consistent environment for program development. It does so by defining controller messages, instrument responses, and message formats for all SCPI compatible instruments. SCPI commands are messages to the instrument to perform specific tasks. The command set includes:

- "SCPI Common Commands" on page 2-2
- "SCPI Required Commands" on page 2-2
- "SCPI Optional Commands" on page 2-2

Note	The Microwave Site Master follows the SCPI standard, but is not fully compliant with that standard. The main reason that the Site Master is not fully compliant is because it does not support all of the required SCPI commands, and because it uses some exceptions in the use of short form and long form command syntax.

SCPI Common Commands

Some common commands are defined in the IEEE-488.2 standard and must be implemented by all SCPI compatible instruments. These commands are identified by the asterisk (*) at the beginning of the command keyword. These commands are defined to control instrument status registers, status reporting, synchronization, and other common functions. For example, `*IDN?` is a common command supported by the Microwave Site Master.

SCPI Required Commands

The required SCPI commands supported by the instrument are listed in the Table 2-1.

Table 2-1. SCPI Required Commands

:STATus
:SYSTem

SCPI Optional Commands

Table 2-2 lists the optional SCPI commands that comprise the majority of the command set described in this document. These commands control most of the programmable functions of the instrument.

Table 2-2. SCPI Optional Commands

:ABORt	:FETCh	:MEASure	:TRACe
:CALCulate	:FORMat	:MMEMory	:TRIGger
:CALibration	:INITiate	:READ	:UNIT
:CONFigure	:INPut	:SENSe	:[SENSe]
:DISPlay	:INSTrument	:SOURce	

The SCPI optional commands are sorted by measurement modes and commands may be repeated in more than one mode.

- Chapter 3, “All Mode Commands”
- Chapter 4, “Cable & Antenna Analyzer Mode Commands”

2-3 Subsystem Commands

Subsystem commands control all instrument functions and some general purpose functions. All subsystem commands are identified by the colon used between keywords, as in `:INITiate:CONTinuous`.

The following information is provided for each subsystem command described in the following chapters.

- The command name, see "Command Names" on page 2-3.
- The path from the subsystem root command, see "Hierarchical Command Structure" on page 2-4.
- The query form of the command (if applicable), see "Query Commands" on page 2-5.
- A description of the purpose of the command.
- The data parameters used as arguments for the command, see "Data Parameters" on page 2-6. This may include the parameter type and the available parameter choices.

Command Names

Typical SCPI commands consist of one or more keywords, parameters, and punctuation. SCPI command keywords can be a mixture of upper and lower case characters. Except for common commands, each keyword has a long and a short form. In this manual, the long form is presented with the short form in upper case and the remainder in lower case. For example, the long form of the command keyword to control the instrument display is `:DISPlay`.

The short form keyword is usually the first four characters of the long form (example: DISP for DISPlay). The exception to this is when the long form is longer than four characters and the fourth character is a vowel. In such cases, the vowel is dropped and the short form becomes the first three characters of the long form. Example: the short form of the keyword `:POWer` is `:POW`.

Some command keywords may have a numeric suffix to differentiate between multiple instrument features such as multiple trace options. For example, `:CALCulate#:DATA? FDATa|SDATa|FMEM|SMEM` can result in two different commands, one for trace 1 `":CALC1:DATA? FDATa"` and another for trace 2 `":CALC2:DATA? FMEM"`.

Note If a numeric suffix is not included in a command, the first option is implied. Curly brackets { } designate optional keyword or command parameters. Square brackets [ ] designate optional command keywords. For example, the command `:TRACe[:DATA]? {1|2}` can be sent as `:TRACe?` or `:TRACe? 1`, or as `:TRAC?` or `:TRAC? 1` to obtain data from trace 1.

As with any programming language, the exact command keywords and command syntax must be used. The syntax of the individual commands is described in detail in the programming command chapters. Unrecognized versions of long form or short form commands, or improper syntax, will generate an error.

Long Format vs. Short Format

Each keyword has a long format and a short format. The start frequency can be specified by `:SENSe:FREQuency:STARt` or `:SENS:FREQ:STAR`. The capital letters in the command specification indicate the short form of the command. A mixture of the entire short form elements with entire long form elements of each command is acceptable. For example, `:SENS:FREQuency:STAR` is an acceptable form of the command. However, `:SENS:FREQuen:STA` is not an acceptable form of the command because `:FREQuen` is not the entire short or long form of the command element.

Hierarchical Command Structure

All SCPI commands, except the common commands, are organized in a hierarchical structure similar to the inverted tree file structure used in most computers. The SCPI standard refers to this structure as "the Command Tree." The command keywords that correspond to the major instrument control functions are located at the top of the command tree. The root command keywords for the SCPI command set are shown in Figure 2-1.

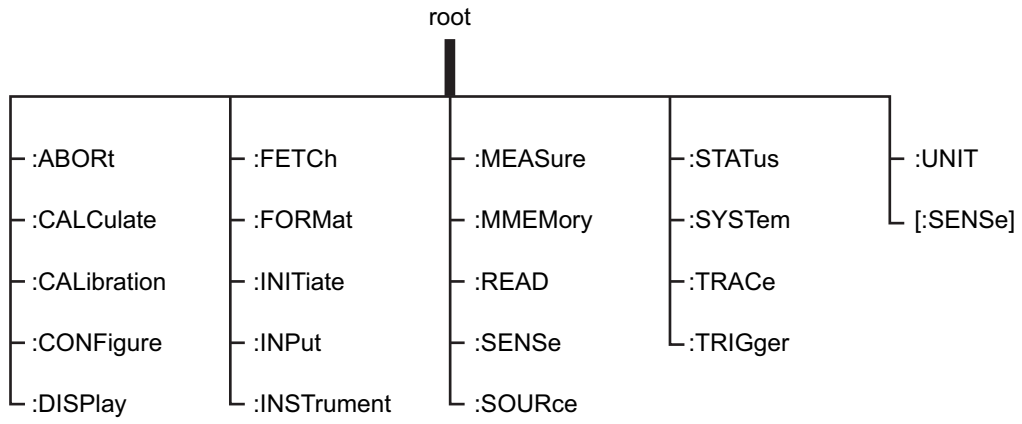


Figure 2-1. SCPI Command Tree

All instrument SCPI commands, except the `:ABORt` command, have one or more subcommands (keywords) associated with them to further define the instrument function to be controlled. The subcommand keywords may also have one or more associated subcommands (keywords). Each subcommand level adds another layer to the command tree. The command keyword and its associated subcommand keywords form a portion of the command tree called a command subsystem. The `:CONFigure` command subsystem is shown in Figure 2-2.

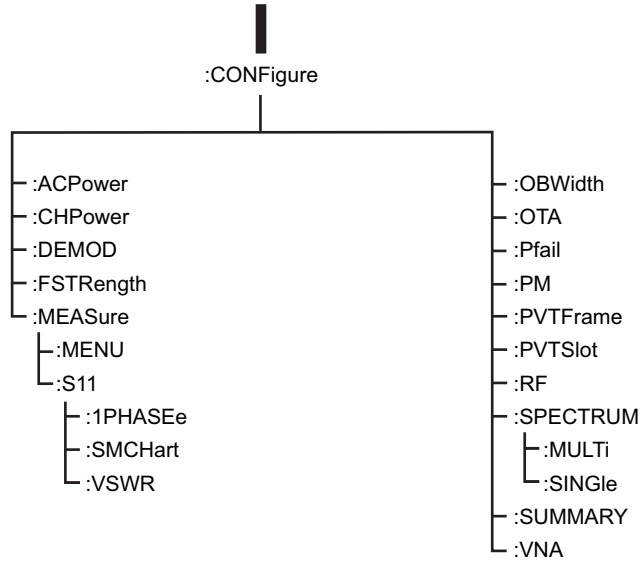


Figure 2-2. SCPI :CONFigure Subsystem

A colon (:) separates each subsystem. For example, the command `:SENSe:FREQuency:STARt <freq>` sets the start frequency. The start frequency is part of the `:FREQuency` subsystem which is part of the `:SENSe subsystem`. Stop frequency is also part of the `:SENSe:FREQuency` subsystem. It is specified by `:SENSe:FREQuency:STOP`.

Query Commands

All commands, unless specifically noted in the commands syntax descriptions, have a query form. As defined in IEEE-488.2, a query is a command with a question mark symbol appended (examples: `*IDN?` and `:OPTions?`). When a query form of a command is received, the current setting associated with the command is placed in the output buffer. Query commands always return the short form of the parameter unless otherwise specified. Boolean values are returned as `1` or `0`, even when they can be set as `on` or `off`.

Data Parameters

Data parameters, referred to simply as “parameters,” are the quantitative values used as arguments for the command keywords. The parameter type associated with a particular SCPI command is determined by the type of information required to control the particular instrument function. For example, Boolean (ON | OFF) type parameters are used with commands that control switch functions.

Some command descriptions specify the type of data parameter to be used with each command. The most commonly used parameter types are numeric, extended numeric, discrete, and Boolean.

Numeric

Numeric parameters comprise integer numbers or any number in decimal or scientific notation, and may include polarity signs.

Discrete

Discrete parameters, such as INTernal and EXTernal, are used to control program settings to a predetermined finite value or condition.

Boolean

Boolean parameters represent binary conditions and may be expressed as ON, OFF or 1, 0. Boolean parameters are always returned by query commands as 1 or 0 in numeric value format.

Unit Suffixes

Unit suffixes are not required for data parameters, provided the values are scaled for the global default units. The instrument SCPI default units are: Hz (Hertz) for frequency related parameters s (seconds) for time related parameters, and m (meters) for distance related parameters.

2-4 Notational Conventions

The SCPI interface standardizes command syntax and style that simplifies the task of programming across a wide range of instrumentation. As with any programming language, the exact command keywords and command syntax must be used. Unrecognized commands or improper syntax will not function.

Table 2-3. Notational Conventions

:	A colon links command keywords together to form commands. The colon is not an actual part of the keyword, but is a signal to the SCPI interface parser. A colon must precede a root keyword immediately following a semicolon (see "Notational Examples" on page 2-8).
;	A semicolon separates commands if multiple commands are placed on a single program line.
[ ]	Square brackets enclose one or more optional keywords.
{ }	Braces enclose one or more keyword or command parameters that may be included zero or more times.
|	A vertical bar indicates "or" and is used to separate alternative parameter options. Example: ON | OFF is the same as ON or OFF.
< >	Angle brackets enclose parameter descriptions.
::=	Means "is defined as" For example: <a>::=<b><c> indicates that <b><c> can replace <a>.
sp	Space, referred to as *white space*, must be used to separate keywords from their associated data parameters. It must not be used between keywords or inside keywords.
xxx	Indicates a root command name
#	Indicates an integer value selection from a range of values

For further information about SCPI command syntax and style, refer to the Standard Commands for Programmable Instruments (SCPI) 1999.0 document.

2-5 Notational Examples

Table 2-4 provides examples of valid command syntax:

Table 2-4. Creating Valid Commands

Command Specification	Valid Forms

`[:SENSe]:FREQuency:STARt <frequency>{Hz|kHz|MHz|GHz}`	The following all produce the same result:
	`:SENSe:FREQuency:STARt 1 MHZ`
	`:SENS:FREQ:STAR 1 MHZ`
	`:sense:frequency:start 1000000`
	`:FREQ:STAR 1000 KHZ`

`:CALCulate:MARKer#:X <value>{Hz|kHz|MHz|GHz,m|cm|mm,ft}`	The first 2 commands set the location of marker 1. The third command sets the location of marker 2.
	`:CALC:MARK:X 1 GHZ`
	`:CALC:MARK1:X 1 GHZ`
	`:CALC:MARK2:X 2 GHZ`

`:INITiate:CONTinuous OFF|ON|0|1`	The following commands are identical:
	`:INITiate:CONTinuous OFF`
	`:init:cont 0`

Command statements read from left to right and from top to bottom. In the command statement above, the `:FREQuency` keyword immediately follows the `:SENSe` keyword with no separating space. A space (*sp*) is used between the command string and its argument.

Note that the first keyword in the command string does not require a leading colon; however, it is good practice to always use a leading colon for all keywords. Note also that the `:SENSe` keyword is optional. This is a SCPI convention for all voltage or signal source type instruments that allows shorter command statements to be used.

The following is an example of a multiple command statement that uses two separate commands in a single statement:

```
:FREQuency:STARt 10E6;:FREQuency:STOP 20E9
```

Note	A semicolon is used to join the commands and a leading colon used immediately after the semicolon to start the second command.

Command Terminators

The <new line> character (ASCII 10) in the last data byte of a command string is used as a command terminator. Use of a command terminator will reset the command path to the root of the tree.

2-6 Formatting Conventions

This manual uses the conventions listed below in describing SCPI commands. The abbreviations “Cmd” and “Param” are used to represent “Command” and “Parameter”.

Table 2-5. Formatting Conventions

`:COMMands:LOOK:LIKE:THIS`	Commands are formatted to differentiate them from their description.
`:COMMand:QUERies:LOOK:LIKE:THIS?`	The query form of the command is followed by a “?”
`<identifier>`	Identifiers are enclosed in “< >”. They indicate that some type of data must be provided.
`|`	The “|” indicates that a choice must be made.
`[optional input]`	Optional input is enclosed in “[ ]”. The “[ ]” are not part of the command.

Chapter 3 — All Mode Commands

The commands in this section apply to all instrument modes.

***IDN?**

Query Description: Provides information about the device. Return format: Anritsu,<model-number>/<options>,<serial-number>,<package-version>

Example: To get device information:

```
*IDN?
```

***OPC?**

Query Description: Blocks the SCPI engine from receiving new commands until system completes all pending operations. Returns 1 for operation complete.

Example: To block until operation is complete:

```
*OPC?
```

:BASe:DIRectory:COPY <"source-directory">,<"destination-directory">

Cmd Description: Copy all files from source directory to destination directory.

Param Type: String, String

Param Description: Source directory, Destination directory

Example: To copy a directory:

```
:BAS:DIR:COPY "/internal/folder1/",
"/internal/folder2/"
```

:BASe:DIRectory:DELete <"directory">, {1|0}

Cmd Description: Delete contents in a directory.

Param Type: String, Integer

Param Description: Param1: Directory path, Optional Param2: Delete mode - 1 to delete the directory and contents, 0 to delete contents only. Default is 0.

Example: To delete the contents of a directory:

```
:BAS:DIR:DEL "/internal/folder1/"
```

:BASe:DIRectory:MAKe <"directory">

Cmd Description: Create a directory.

Param Type: String

Param Description: Directory path

Example: To create a directory:

```
:BAS:DIR:MAK "/internal/folder1/"
```

:BASe:FILe:COPY <"source-filename">, <"destination-filename">

Cmd Description: Copy an existing file to another filepath. For USB drives, ensure /usb/ or /usb2/ is correctly used as the root folder.

Param Type: String, String

Param Description: Param1: Source filename, Param2: Destination filename. The folder structure must exist before copying

Example: To copy a file to a connected USB drive:

```
:BAS:FIL:COPY "/internal/Temp.dat",
"/usb/CopiedFile.dat"
```

:BASe:FILe:DELete <"filename">

Cmd Description: Delete a file.

Param Type: String

Param Description: Filename with extension

Example: To delete a file:

```
:BAS:FIL:DEL "/internal/temp.dat"
```

:CONFigure:BASe:FACTorydefault

Cmd Description: This command sets system to factory default state.

Example: To set to factory default:

```
:CONF:BAS:FACT
```

:CONFigure:BASe:MASTerreset

Cmd Description: This command sets system to master reset state.

Example: To perform master reset:

```
:CONF:BAS:MAST
```

:CONFigure:BASe:POWeroff 0|1

Cmd Description: Command cycles the power to the instrument (0) or turns it completely off (1). Cycling the power does not save the current setup (instrument will restart with last saved setup) whereas turning the instrument off will store the current setup.

Param Range: 0 for power cycling, 1 for turning instrument off

Example: To cycle the system power:

```
:CONF:BAS:POW 0
```

:CONFigure:VIP:ANALyze:AUTO ON|OFF|1|0
:CONFigure:VIP:ANALyze:AUTO?

Cmd Description: Command sets auto analyze state.

Query Description: Query returns auto analyze state. Query returns 1 if auto analyze is on. Otherwise, it returns 0.

Param Range: ON|OFF|1|0

Default Value: OFF|0

Example: To set auto analyze state to on:

```
:CONF:VIP:ANAL:AUTO ON
```

:FETCh:GPSData:RESet

Cmd Description: Reset the current and last GPS data.

Example: To reset GPS data:

```
:FETC:GPSD:RES
```

:FETCh:GPSData? {CURRent|LAST}

Query Description: Query returns the current or the last GPS data. If no parameter is entered, it defaults to return current GPS data. Latitude and longitude units are in radians. Altitude units are in meters. Return format: No H/W Found|No Fix|Good Fix (3D),<time-stamp>,<latitude>,<longitude>,<altitude>|Good Fix (2D),<time-stamp>,<latitude>,<longitude>

Param Description: Optional Param: GPS data collection type, default is CURRent.

Param Range: CURRent|LAST

Example: Query the current GPS data:

```
:FETC:GPSD? CURRent
```

:INSTrument:CATalog:FULL?

Query Description: Queries the available modes. 'CAA'2 for Advanced Cable Antenna Analyzer mode, 'Power Meter'6 for Power Meter, 'HI_PM'10 for High Accuracy Power Meter, 'cCAA'201 for Classic Cable Antenna Analyzer mode, 'VIP'202 for Visual Inspection Probe

Example: To query available application modes:

```
:INST:CAT:FULL?
```

:INSTrument:NSELect <application-mode-number>
:INSTrument:NSELect?

Cmd Description: Command sets the required application mode.

Query Description: Query returns which mode is currently selected. 2 for Advanced Cable Antenna Analyzer mode, 6 for Power Meter,10 for High Accuracy Power Meter, 201 for Classic Cable Antenna Analyzer mode, 202 for Visual Inspection Probe.

Param Type: Integer

Param Description: Application Mode

Param Range: 2|6|10|201|202

Example: To set current application mode to Hi Accuracy Power Meter:

```
:INST:NSEL 10
```

:INSTrument:SELect <"mode-identifier">
:INSTrument:SELect?

Cmd Description: Command sets the required application mode.

Query Description: Query returns which mode is currently selected. "CAA" for Advanced Cable Antenna Analyzer mode, "Power Meter" for Power Meter, "HI_PM" for High Accuracy Power Meter, "cCAA" for Classic Cable Antenna Analyzer mode, "VIP" for Visual Inspection Probe

Param Type: String

Param Description: Application mode identifier

Param Range: "CAA"|"Power Meter"|"HI_PM"|"cCAA"|"VIP"

Example: To set current application mode to Hi Accuracy Power Meter:

```
:INST:SEL "HI_PM"
```

:MMEMory:DELete <"filename">

Cmd Description: Delete setup(.stp) or measurement(.dat) file.

Param Type: String

Param Description: Filepath with extension

Example: To delete a setup:

```
:MMEM:DEL "/internal/temp.stp"
```

:MMEMory:LOAD:STATe 1,<"filename">

Cmd Description: Recall the given setup file.

Param Type: Integer, String

Param Description: Param1: Must send 1 as first parameter, Param2: Filepath with extension.

Example: To recall a setup (include extension):

```
:MMEM:LOAD:STAT 1, "/internal/temp.stp"
```

:MMEMory:LOAD:TRACe 1,<"filename">

Cmd Description: Recall the given measurement file.

Param Type: Integer, String

Param Description: Param1: Must send 1 as first parameter, Param2: Filepath with extension.

Example: To recall a measurement (include extension):

```
:MMEM:LOAD:TRAC 1, "/internal/temp.dat"
```

:MMEMory:STORe:PNG 0,<"filename">

Cmd Description: Save screenshot as a .png file.

Param Type: Integer, String

Param Description: Param1: Must send 0 as first parameter, Param2: Filepath without extension.

Example: To save a screenshot (include directory path, do not include extension):

```
:MMEM:STOR:PNG 0, "/internal/temp"
```

:MMEMory:STORe:STATe 0,<"filename">

Cmd Description: Save the user settings as a setup file.

Param Type: Integer, String

Param Description: Param1: Must send 0 as first parameter, Param2: Filepath without extension.

Example: To save a setup (include directory path, do not include extension):

```
:MMEM:STOR:STAT 0, "/internal/temp"
```

Notes: In VIP mode, use this command to save .vipi file. Image must be captured first in order for save to happen.

:MMEMory:STORe:TRACe <integer>,<"filename">

Cmd Description: Stores the trace data and settings into a file with specified type.

Param Type: Integer, String

Param Description: Param1: An integer representing the file type where 1 or 0 : Measurement file (default), Param2: Filepath without extension.

Example: To save a measurement (include directory path, do not include extension):

```
:MMEM:STOR:TRAC 0, "/internal/temp"
```

:PROGram:ETT:ABORtscript

Cmd Description: Command exits easy-test mode.

Example: To abort ETT mode:

```
:PROG:ETT:ABOR
```

:PROGram:ETT:LOADscript <"filename">

Cmd Description: Command begins easy-test mode with the given filename.

Param Description: Filename with .ett extension

Example: To enter ETT mode:

```
:PROG:ETT:LOAD "/internal/my-test.ett"
```

:PROGram:ETT:NEXTstep

Cmd Description: Command performs the next step while in easy-test mode.

Example: To proceed to the next step in ETT mode:

```
:PROG:ETT:NEXT
```

:PROGram:ETT:STATe?

Example: To get the ETT state:

```
:PROG:ETT:STAT?
```

:SYSTem:CODeload

Cmd Description: Command performs a firmware update from a USB drive.

Example: To initiate a firmware update:

```
:SYST:COD
```

:SYSTem:MBTemperature?

Query Description: Query returns motherboard temperature in degree C.

Example: To query motherboard temperature:

```
:SYST:MBT?
```

:SYSTem:OPTions?

Query Description: Query returns option string delimited by "/".

Example: To query full option string:

```
:SYST:OPT?
```

:SYSTem:PRESet

Cmd Description: Presets the application to its default state. Use *OPC? to block till preset is complete.

Example: To perform preset:

```
:SYST:PRES
```

[:SENSe]:BASe:BRIGhtness <brightness-level>
[:SENSe]:BASe:BRIGhtness?

Cmd Description: Command sets the brightness level.

Query Description: Query returns the brightness level.

Param Type: Integer

Param Range: 0 to 10 in increments of 1, where 0 is 0% and 10 is 100% brightness setting.

Default Value: 10

Example: To set brightness to 5:

```
:BAS:BRIG 5
```

[:SENSe]:BASe:IPaddress?

Query Description: Query returns the IP address in a special format. Return value example: 2887436918 (to recover the actual IP address of 172.26.202.118, you need to convert the returned value to hex AC1ACA76 and then take the resulting 4 numbers (2 characters each, in hex) and convert them back to decimal.

Example: To query IP address:

```
:BAS:IP?
```

[:SENSe]:BASe:NET:MANual:GATEway <"Default-Gateway">
[:SENSe]:BASe:NET:MANual:GATEway?

Cmd Description: Command sets the static Gateway.

Query Description: Query returns the static Gateway in string format.

Param Type: String

Param Description: Default Gateway

Example: To set static Gateway:

```
:BAS:NET:MAN:GATE "192.35.101.1"
```

[:SENSe]:BASe:NET:MANual:IP <"IP-address">
[:SENSe]:BASe:NET:MANual:IP?

Cmd Description: Command sets the static IP address.

Query Description: Query returns the static IP address in string format.

Param Type: String

Param Description: IP address

Example: To set static IP address:

```
:BAS:NET:MAN:IP "192.35.100.1"
```

[:SENSe]:BASe:NET:MANual:SUBnet <"Subnet-Mask">
[:SENSe]:BASe:NET:MANual:SUBnet?

Cmd Description: Command sets the static Subnet Mask.

Query Description: Query returns the static Subnet Mask in string format.

Param Type: String

Param Description: Subnet Mask

Example: To set static Subnet Mask:

```
:BAS:NET:MAN:SUB "255.255.252.0"
```

[:SENSe]:BASe:NET:MODE MANual|DHCP
[:SENSe]:BASe:NET:MODE?

Cmd Description: Command sets the Ethernet configuration to DHCP or static (manual).

Query Description: Query returns the Ethernet configuration.

Param Type: Character

Param Description: MANual|DHCP

Example: To set the Ethernet configuration to static (manual):

```
:BAS:NET:MODE MAN
```

[:SENSe]:BASe:NET:MODE:MANual:RESTart

Cmd Description: Command sets the Ethernet configuration to static (manual) and restarts the instrument.

Example: To set the Ethernet configuration to static (manual) and trigger a restart:

```
:BAS:NET:MODE:MAN:REST
```

[:SENSe]:BASe:REMote 0|1|2

Cmd Description: Command sets instrument into local lock-out for remote operation. Send 0 is to get out of remote mode and restore front-panel operation, 1 to set remote mode with synchronous communication, and 2 to set remote mode with asynchronous communication.

Param Type: Integer

Param Range: 1|0

Default Value: 0

Example: To set the instrument to remote synchronous mode and perform a local lock-out:

```
:BAS:REM 1
```

[:SENSe]:BASe:SCRNshot:SCHeme INVerted|STANdard
[:SENSe]:BASe:SCRNshot:SCHeme?

Cmd Description: Command sets the screenshot setting scheme.

Query Description: Query returns the value that represents the current screenshot scheme.

Param Type: Character

Param Range: INVerted|STANdard

Default Value: INVerted

Example: To set the screen capture scheme to standard:

```
:BAS:SCRN:SCH STAN
```

[:SENSe]:BASE:TEMPerature?

Query Description: Query returns motherboard temperature in degree C. Same as :SYSTem:MBTemperature?

Example: To query the motherboard temperature:

```
:BAS:TEMP?
```

[:SENSe]:BASe:TIME <time-in-seconds>
[:SENSe]:BASe:TIME?

Cmd Description: Command sets the time. Send a -1 to synchronize the system time with GPS.

Query Description: Query returns value that represents the number of seconds from 1/1/1970.

Param Type: Integer

Param Description: Number of seconds from 1/1/1970

Default Units: s(seconds)

Example: To set the time to 3:09 on January 5, 2012:

```
:BAS:TIME 1325776177
```

[:SENSe]:BASe:VOLume <volume-level>
[:SENSe]:BASe:VOLume?

Cmd Description: Command sets the volume level.

Query Description: Query returns the volume level.

Param Type: Integer

Param Range: 0 to 10 in increments of 1, where 0 is 0% and 10 is 100% of volume setting.

Default Value: 5

Example: To set volume to 7:

```
:BAS:VOL 7
```

[:SENSe]:BASe:VOLume:AUDio ON|OFF|1|0

Cmd Description: Command sets the audio state. ON means audio is enabled. OFF means audio is disabled.

Param Type: Character | Integer

Param Range: ON | OFF | 1 | 0

Default Value: ON | 1

Example: To mute audio:

```
:BAS:VOL:AUD OFF
```

Chapter 4 — Cable & Antenna Analyzer Mode Commands

The commands in this section apply to the Cable and Antenna Analysis mode.

:CALCulate#:DATA? FDATa|SDATa|FMEM|SMEM

Query Description: Query returns formatted data block of the specified trace.

Param Range: FDATa | SDATa | FMEM | SMEM

Example: To query trace 1 S-data:

```
:CALC1:DATA? SDATA
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode. FDATa: Formatted (or Final) data. The returned data are based on the Measurement Mode that is associated with the trace. For measurement types that use only one number per point (such as Return Loss, VSWR, Cable Loss, Transmission, Phase), the command returns one number per data point. For graph types that use two numbers per point (such as Smith Chart), the command returns two numbers per data point. SDATa: Complex measurement data. The returned numbers, which are independent of the Measurement Mode that is associated with the trace, are complex measurement data pairs (Real and Imaginary) for each point of the trace. A 517 point trace therefore has a total of 1034 values that get transferred. FMEM: Formatted (or Final) Memory data. Similar to FDATa, but for memory data. SMEM: Complex memory data. Similar to SDATa, but for memory data. Note that in order to get valid data when querying for memory data, you must first store a trace into memory. The format of the block data that is returned can be specified by the command :FORMat[:READings][:DATA]. The response begins with an ASCII header that specifies the number of data bytes. It appears in the format #AX, where A is the number of digits in X, and X is the number of bytes that follow the header. Each data point is separated by a comma delimiter.

:CALCulate#:LIMit:LOWer:SEGMent:ADD <StartX><StopX><StartY><StopY>

Cmd Description: Command adds a segment to the lower limit line.

Param Type: Float

Default Units: X units: Hz for Frequency domain, m or ft for Distance domain; Y units: same as current active trace y-axis units

Example: To add a lower limit segment from 1MHz to 1GHz with value 20 dB for trace 1:

```
:CALC1:LIM:LOW:SEGM:ADD 1e6,1e9,20,20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:DELete

Cmd Description: Command deletes the active lower segment.

Example: To delete the active lower segment for trace 1

```
:CALC1:LIM:LOW:SEGM:DEL
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:EDIT <StartX><StopX><StartY><StopY>

Cmd Description: Command edits the lower limit active segment. The active segment index must be updated when adding/deleting a segment.

Param Type: Float

Default Units: X units: Hz for Frequency domain, m or ft for Distance domain; Y units: same as current active trace y-axis units

Example: To edit the active segment and set it to 1MHz to 1GHz with value 20 dB for trace 1:

```
:CALC1:LIM:LOW:SEGM:EDIT 1e6,1e9,20,20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:STARt:X <value>
:CALCulate#:LIMit:LOWer:SEGMent:STARt:X?

Cmd Description: Command sets the lower limit line X start value for current segment.

Query Description: Query returns the lower limit line X start value for active segment.

Param Type: Float

Default Units: Hz for Frequency domain, m or ft for Distance domain.

Example: To set limit X start value to 1 MHz for trace 1:

```
:CALC1:LIM:LOW:SEGM:STAR:X 1000000
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:STARt:Y <amplitude>
:CALCulate#:LIMit:LOWer:SEGMent:STARt:Y?

Cmd Description: Command sets the lower limit line Y start value for active segment.

Query Description: Query returns the lower limit line Y start value for active segment.

Param Type: Float

Default Units: Current active trace y-axis units

Example: To set limit start amplitude to 20 dB for trace 1:

```
:CALC1:LIM:LOW:SEGM:STAR:Y 20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:STOP:X <value>
:CALCulate#:LIMit:LOWer:SEGMent:STOP:X?

Cmd Description: Command sets the lower limit line X stop value for active segment.

Query Description: Query returns the upper limit line X stop value for active segment.

Param Type: Float

Default Units: Hz for Frequency domain, m or ft for Distance domain.

Example: To set limit X stop value to 1 MHz for trace 1:

```
:CALC1:LIM:LOW:SEGM:STOP:X 1000000
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:STOP:Y <amplitude>
:CALCulate#:LIMit:LOWer:SEGMent:STOP:Y?

Cmd Description: Command sets the lower limit line Y stop value for active segment.

Query Description: Query returns the lower limit line Y stop value for active segment.

Param Type: Float

Default Units: Current active trace y-axis units

Example: To set limit stop amplitude to 20 dB for trace 1:

```
:CALC1:LIM:LOW:SEGM:STOP:Y 20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:SEGMent:TOTal?

Query Description: Query returns the total segments in the lower limit for the given trace.

Example: To query how many lower limit segments are currently available for trace 1:

```
:CALC:LIM:LOW:SEGM:TOT?
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer:Y <amplitude>{dB,s|ms|ns|ps}
:CALCulate#:LIMit:LOWer:Y?

Cmd Description: Command sets the lower limit line value. In group delay mode, the value is in units of time.

Query Description: Query returns the lower limit line value.

Param Range: x[,dB]

Default Units: Current active trace y-axis units

Example: To set limit amplitude to 20 dB for trace 1:

```
:CALC1:LIM:LOW:Y 20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:LOWer[:STATe] OFF|ON|0|1
:CALCulate#:LIMit:LOWer[:STATe]?

Cmd Description: Command sets the low limit line state for the specified trace.

Query Description: Query returns the lower limit line state for the specified trace.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set limit on for trace 1:

```
:CALC1:LIM:LOW ON
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:ADD <StartX><StopX><StartY><StopY>

Cmd Description: Command adds a segment to the upper limit line.

Param Type: Float

Default Units: X units: Hz for Frequency domain, m or ft for Distance domain; Y units: same as current active trace y-axis units

Example: To add an upper limit segment from 1MHz to 1GHz with value 20 dB for trace 1:

```
:CALC1:LIM:UPP:SEGM:ADD 1e6,1e9,20,20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:DELete

Cmd Description: Command deletes the active upper segment.

Example: To delete the active upper segment for trace 1

```
:CALC1:LIM:UPP:SEGM:DEL
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:EDIT <StartX><StopX><StartY><StopY>

Cmd Description: Command edits the upper limit active segment. The active segment index must be updated when adding/deleting a segment.

Param Type: Float

Default Units: X units: Hz for Frequency domain, m or ft for Distance domain; Y units: same as current active trace y-axis units

Example: To edit the active segment and set it to 1MHz to 1GHz with value 20 dB for trace 1:

```
:CALC1:LIM:UPP:SEGM:EDIT 1e6,1e9,20,20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:STARt:X <value>
:CALCulate#:LIMit:UPPer:SEGMent:STARt:X?

Cmd Description: Command sets the upper limit line X start value for active segment.

Query Description: Query returns the upper limit line X start value for active segment.

Param Type: Float

Default Units: Hz for Frequency domain, m or ft for Distance domain.

Example: To set limit X start value to 1 MHz for trace 1:

```
:CALC1:LIM:UPP:SEGM:STAR:X 1000000
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:STARt:Y <amplitude>
:CALCulate#:LIMit:UPPer:SEGMent:STARt:Y?

Cmd Description: Command sets the upper limit line Y start value for active segment.

Query Description: Query returns the upper limit line Y start value for active segment.

Param Type: Float

Default Units: Current active trace y-axis units

Example: To set limit start amplitude to 20 dB for trace 1:

```
:CALC1:LIM:UPP:SEGM:STAR:Y 20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:STOP:X <value>
:CALCulate#:LIMit:UPPer:SEGMent:STOP:X?

Cmd Description: Command sets the upper limit line X stop value for active segment.

Query Description: Query returns the upper limit line X stop value for active segment.

Param Type: Float

Default Units: Hz for Frequency domain, m or ft for Distance domain.

Example: To set limit X stop value to 1 MHz for trace 1:

```
:CALC1:LIM:UPP:SEGM:STOP:X 1000000
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:STOP:Y <amplitude>
:CALCulate#:LIMit:UPPer:SEGMent:STOP:Y?

Cmd Description: Command sets the upper limit line Y stop value for active segment.

Query Description: Query returns the upper limit line Y stop value for active segment.

Param Type: Float

Default Units: Current active trace y-axis units

Example: To set limit stop amplitude to 20 dB for trace 1:

```
:CALC1:LIM:UPP:SEGM:STOP:Y 20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:SEGMent:TOTal?

Query Description: Query returns the total segments in the upper limit for the given trace.

Example: To query how many upper limit segments are currently available for trace 1:

```
:CALC:LIM:UPP:SEGM:TOT?
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer:Y <amplitude>{dB,s|ms|ns|ps}
:CALCulate#:LIMit:UPPer:Y?

Cmd Description: Command sets the upper limit line value. In group delay mode, the value is in units of time.

Query Description: Query returns the upper limit line value.

Param Type: Float

Default Units: Current active trace y-axis units

Example: To set limit amplitude to 20 dB for trace 1:

```
:CALC1:LIM:UPP:Y 20
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate#:LIMit:UPPer[:STATe] OFF|ON|0|1
:CALCulate#:LIMit:UPPer[:STATe]?

Cmd Description: Command sets the upper limit line state for the specified trace.

Query Description: Query returns the upper limit line state for the specified trace.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set limit on for trace 1:

```
:CALC1:LIM:UPP ON
```

Notes: Replace '#' with the required trace number. Check User Guide for the number of supported traces in each mode.

:CALCulate:LIMit:ALARm OFF|ON|0|1
:CALCulate:LIMit:ALARm?

Cmd Description: Command sets the limit alarm state. The device beeps if a trace point crosses the limit line.

Query Description: Query returns the limit alarm state.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set limit alarm on:

```
:CALC:LIM:ALAR ON
```

:CALCulate:LIMit:LOWer:SEGMent:ACTive <Index>

Cmd Description: Command sets the given lower segment index active.

Param Type: int

Example: To set lower segment 1 active:

```
:CALC:LIM:LOW:SEGM:ACT 1
```

:CALCulate:LIMit:MESSage OFF|ON|0|1
:CALCulate:LIMit:MESSage?

Cmd Description: Command sets the limit message state. Displays a Pass/Fail message to indicate whether the trace crosses the limit line.

Query Description: Query returns the limit message state.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set limit message on:

```
:CALC:LIM:MESS ON
```

:CALCulate:LIMit:MESSage:RESult?

Query Description: Query returns 0 if limit line says Fail. Otherwise, it returns 1 if limit line says Pass.

Example: To get limit line Pass/Fail result:

```
:CALC:LIM:MESS:RES?
```

:CALCulate:LIMit:MODe

Cmd Description: Command changes the limit line to either Line or Segmented mode.

Example: To change the limit line mode

```
:CALC:LIM:MOD SEGM
```

:CALCulate:LIMit:PRESet

Cmd Description: Command presets the limit line to its default state.

Example: To perform limit line preset:

```
:CALC:LIM:PRES
```

:CALCulate:LIMit:TYPE 0|1
:CALCulate:LIMit:TYPE?

Cmd Description: Command sets the active limit line type. 0 for upper limit and 1 for lower limit.

Query Description: Query returns the active limit line type.

Param Range: 0|1

Default Value: 0

Example: To set the active limit to lower limit:

```
:CALC:LIM:TYPE 1
```

:CALCulate:LIMit:UPPer:SEGMent:ACTive <Index>

Cmd Description: Command sets the given upper segment index active.

Param Type: int

Example: To set upper segment 1 active:

```
:CALC:LIM:UPP:SEGM:ACT 1
```

:CALCulate:LIMit:VALue <amplitude>{dB}
:CALCulate:LIMit:VALue?

Cmd Description: Command sets the limit line value.

Query Description: Query returns the limit line value.

Param Type: Float

Default Units: Current active trace y-axis units

Example: To set limit line to 20 dB:

```
:CALC:LIM:VAL 20
```

:CALCulate:LIMit[:STATe] OFF|ON|0|1
:CALCulate:LIMit[:STATe]?

Cmd Description: Command sets the limit line state.

Query Description: Query returns the limit line state.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set limit on:

```
:CALC:LIM ON
```

:CALCulate:MARKer#:PEAK

Cmd Description: This command sets specified marker to the trace peak value.

Example: To set marker 5 to the peak of the trace:

```
:CALC:MARK5:PEAK
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode.

:CALCulate:MARKer#:PEAK:BOUNded

Cmd Description: This command sets specified marker to the trace peak value between two markers. Markers 5 and 7 will search for the peak between markers 1 and 2. Markers 6 and 8 will search for the peak between markers 3 and 4.

Example: To set marker 5 to the peak between markers 1 and 2:

```
:CALC:MARK5:PEAK:BOUN
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode. If any marker other than 5, 6, 7, or 8 is specified, the bounded marker will be set to 5 by default.

:CALCulate:MARKer#:TRACking PEAK|VALLey|BNDPeak|BNDValley|OFF
:CALCulate:MARKer#:TRACking?

Cmd Description: Command sets the tracking mode of the specified marker.

Query Description: Query returns the tracking mode of the specified marker. Query returns PEAK for peak tracking, or returns VALL for valley tracking, or returns BNDP for bounded peak tracking, or returns BNDV for bounded valley tracking.

Param Range: PEAK | VALLey | BNDPeak | BNDValley | OFF

Default Value: OFF

Example: To set marker 2 to peak tracking:

```
:CALC:MARK2:TRAC PEAK
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode. If any marker other than 5, 6, 7, or 8 is specified, the bounded marker will be set to 5 by default.

:CALCulate:MARKer#:TYPE REFerence|DELTa
:CALCulate:MARKer#:TYPE?

Cmd Description: Command sets the type of the specified marker.

Query Description: Query returns the type of the specified marker. Query returns REF if the specified marker is a reference marker, or returns DELT if the specified marker is a delta marker.

Param Range: REFerence | DELTA

Default Value: REF

Example: To set marker 2 type to delta:

```
:CALC:MARK2:TYPE DELT
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode.

:CALCulate:MARKer#:VALLey

Cmd Description: This command sets specified marker to the trace valley value.

Example: To set marker 5 to the valley of the trace:

```
:CALC:MARK5:VALL
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode.

:CALCulate:MARKer#:VALLey:BOUNded

Cmd Description: This command sets specified marker to the trace valley value between two markers. Markers 5 and 7 will search for the valley between markers 1 and 2. Markers 6 and 8 will search for the valley between markers 3 and 4.

Example: To set marker 5 to the valley between markers 1 and 2:

```
:CALC:MARK5:VALL:BOUN
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode. If any marker other than 5, 6, 7, or 8 is specified, the bounded marker will be set to 5 by default.

:CALCulate:MARKer#:X <value>{Hz|kHz|MHz|GHz,m|cm|mm,ft}
:CALCulate:MARKer#:X?

Cmd Description: Command sets the position for the specified marker in the current sweep domain. For delta marker, it sets the relative position to the reference marker. Verify that the appropriate unit is used.

Query Description: Query returns the marker position for the specified marker. For a delta marker, query returns the position relative to the reference marker.

Param Type: Float

Param Description: Frequency, or distance.

Default Value: Model-dependent

Default Units: Hz

Example: To set marker 2 frequency to 1 GHz:

```
:CALC:MARK2:X 1 GHZ
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode.

:CALCulate:MARKer#:Y?

Query Description: Query returns the amplitude value for the specified marker.

Example: To query marker 3 amplitude value:

```
:CALC:MARK3:Y?
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode.

:CALCulate:MARKer#[:STATe] OFF|ON|0|1
:CALCulate:MARKer#[:STATe]?

Cmd Description: Command sets the state of the specified marker.

Query Description: Query returns the state of the specified marker. Query returns 1 if the specified marker is on, or returns 0 if the specified marker is off.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set marker 1 state ON:

```
:CALC:MARK1 ON
```

Notes: Replace '#' with the required marker number. Check User Guide for the number of supported markers in each mode.

:CALCulate:MARKer:ACTive?

Query Description: Query returns active marker number. Returns NONE if no markers are active.

Example: To get active marker

```
:CALC:MARK:ACT?
```

:CALCulate:MARKer:DISPlay MKRTable|MKRonly|AOFF
:CALCulate:MARKer:DISPlay?

Cmd Description: Command sets the marker display mode.

Query Description: Query returns the marker display mode. Query returns MKRT if the markers and marker table are displayed, returns MKR if only the markers are displayed, or returns AOFF if markers and marker table are not displayed.

Param Range: MKRTable|MKRonly|AOFF

Default Value: MKRTable

Example: To set marker display to marker only:

```
:CALC:MARK:DISP MKR
```

:CALCulate:MARKer:PRESet

Cmd Description: Presets markers.

Example: To preset markers:

```
:CALC:MARK:PRES
```

:CALCulate:MATH:FUNCtion NONE|SUBTraction|ADDition|AVERage
:CALCulate:MATH:FUNCtion?

Cmd Description: Command sets the trace math function. NONE - no trace math applied; SUBT - memory trace subtracted from active trace; ADD - memory trace added to active trace; AVER - active and memory traces added and divided by 2.

Query Description: Query returns the trace math function.

Param Range: NONE | SUBTraction | ADDition | AVERage

Default Value: NONE

Example: To set trace math function to addition:

```
:CALC:MATH:FUNC ADD
```

:CALCulate:MATH:MEMorize

Cmd Description: Copies active trace into the memory buffer.

Example: To copy active trace into the memory buffer:

```
:CALC:MATH:MEM
```

:CALCulate:TRANsform:CLAVerage?

Query Description: Query returns cable loss average when the measurement mode is 'Cable Loss'.

Default Units: dB

Example: To query cable loss average:

```
:CALC:TRAN:CLAV?
```

:CALCulate:TRANsform:DISTance:CABLoss <cable-loss>{dB/m,dB/ft}
:CALCulate:TRANsform:DISTance:CABLoss?

Cmd Description: Command sets the cable loss. Verify that the appropriate unit is used.

Query Description: Query returns the cable loss based on the current units.

Param Type: Float

Default Value: 0

Default Units: dB/m

Example: To set cable loss to 2:

```
:CALC:TRAN:DIST:CABL 2
```

:CALCulate:TRANsform:DISTance:DMAX?

Query Description: Query returns max distance in the current units.

Example: To query max distance:

```
:CALC:TRAN:DIST:DMAX?
```

:CALCulate:TRANsform:DISTance:PVELocity <propagation-velocity>
:CALCulate:TRANsform:DISTance:PVELocity?

Cmd Description: Command sets the propagation velocity.

Query Description: Query returns the propagation velocity.

Param Type: Float

Default Value: 1

Example: To set propagation velocity to 0.05:

```
:CALC:TRAN:DIST:PVEL 0.05
```

:CALCulate:TRANsform:DISTance:RESolution?

Query Description: Query returns distance resolution in the current units for reflection.

Example: To query distance resolution:

```
:CALC:TRAN:DIST:RESolution?
```

:CALCulate:TRANsform:DISTance:STARt <distance>{ m|cm|mm,ft}
:CALCulate:TRANsform:DISTance:STARt?

Cmd Description: Command sets the start distance. Verify the appropriate unit is used.

Query Description: Query returns the start distance in the current units.

Param Type: Float

Param Description: Distance

Default Units: m(meter)

Example: To set start distance to 1 m:

```
:CALC:TRAN:DIST:STAR 1
```

:CALCulate:TRANsform:DISTance:STOP <distance>{ m|cm|mm,ft}
:CALCulate:TRANsform:DISTance:STOP?

Cmd Description: Command sets the stop distance. Verify the appropriate unit is used.

Query Description: Query returns the stop distance in the current units.

Param Type: Float

Param Description: Distance

Default Units: m(meter)

Example: To set stop distance to 2 m:

```
:CALC:TRAN:DIST:STOP 2 M
```

:CALCulate:TRANsform:DISTance:UNIT METers|FEET
:CALCulate:TRANsform:DISTance:UNIT?

Cmd Description: Command sets the distance unit.

Query Description: Query returns the current distance units.

Param Range: METers|FEET

Default Value: METers

Example: To set distance unit to feet:

```
:CALC:TRAN:DIST:UNIT FEET
```

:CALCulate:TRANsform:DISTance:WINDow
RECTangular|MSLobe|NSLobe|LSLobe
:CALCulate:TRANsform:DISTance:WINDow?

Cmd Description: Command sets the window type for distance domain.

Query Description: Query returns the window type for distance domain.

Param Type: Character

Param Range: RECTangular | MSLobe | NSLobe | LSLobe

Default Value: NSLobe

Example: To set distance domain window type to rectangular:

```
:CALC:TRAN:DIST:WIND RECT
```

:CAPTure:STARt

Cmd Description: Command starts the data capture for continuous sweeps. It does not work in single sweep and hold mode.

Example: To start data capture

```
:CAPT:STAR
```

:CAPTure:STOP

Cmd Description: Command stops the data capture for continuous sweeps. It does not work in single sweep and hold mode.

Example: To stop capture:

```
:CAPT:STOP
```

:CAPTure:TRACe#:DATA? FDATa|SDATa|FMEM|SMEM

Cmd Description: Command is used to return the data block of the specified trace at the end of a sweep, while the instrument is in continuous sweep mode. This command must follow the :CAPTure:STARt command. A :CAPTure:STOP command is used to end the capture session. If the system is in single sweep and hold mode, then this command is ignored.

Example: To trigger a sweep:

```
:CAPT:TRAC1:DATA? FDAT
```

:CONFigure:MEASure:DISPlay SINGle|DUAL
:CONFigure:MEASure:DISPlay?

Cmd Description: Command sets the display format to single or dual traces. Only supported in Advanced Cable & Antenna Analyzer mode.

Query Description: Query returns the display format. Only supported in Advanced Cable & Antenna Analyzer mode.

Param Range: SINGle|DUAL

Default Value: SINGle

Example: To set display format to dual traces:

```
:CONF:MEAS:DISP DUAL
```

:CONFigure:MEASure:MODe
RLFReq|RLDTf|SWRFreq|SWRDtf|CLFReq|TRES|SMITh|PHASe
:CONFigure:MEASure:MODe?

Cmd Description: Command sets the active trace measurement mode.

Query Description: Query returns the current measurement mode.

Param Description: RLFReq: Return Loss(Frequency domain sweep); RLDTf: Return Loss(DTF); SWRFreq: SWR(Frequency domain sweep); SWRDtf: SWR(DTF); CLFReq: Cable Loss - One Port(Frequency domain sweep); TRES: Transmission - External Sensor; SMITh: Smith Chart; PHASe: One-Port Phase. TRES, SMITh, and PHASe are not available in Classic Mode.

Example: To set measurement mode to Cable Loss:

```
:CONF:MEAS:MOD CLFR
```

:DISPlay:UPDate ON|OFF|1|0
:DISPlay:UPDate?

Cmd Description: Command sets the display update on/off.

Query Description: Query returns display update status. Query returns 1 if display update is on. Otherwise, it returns 0.

Param Range: ON|OFF|1|0

Default Value: OFF|0

Example: To turn display update off:

```
:DISP:UPD OFF
```

:DISPlay:WINDow:TRACe:STATe TRACe|MEMory|BOTH
:DISPlay:WINDow:TRACe:STATe?

Cmd Description: Command sets the trace display type for the active trace. TRAC - only active trace is shown; MEM - only memory trace is shown; BOTH - both active and memory traces are shown. Use :CALCulate:MATH:MEMorize before changing the display type.

Query Description: Query returns the trace display type.

Param Range: TRACe | MEMory | BOTH

Default Value: TRACe

Example: To show both active and memory trace on the display:

```
:DISP:WIND:TRAC:STAT BOTH
```

:DISPlay:WINDow:TRACe:Y[:SCALe]:AUToscale

Cmd Description: This command sets the top and bottom values such that the trace occupies a significant portion of the grid.

Example: To perform auto-scale:

```
:DISP:WIND:TRAC:Y:AUT
```

:DISPlay:WINDow:TRACe:Y[:SCALe]:BOTTom <amplitude>{dB}
:DISPlay:WINDow:TRACe:Y[:SCALe]:BOTTom?

Cmd Description: Command sets the display bottom value.

Query Description: Query returns the display bottom value.

Param Type: Float

Default Value: Return Loss = 60dB; DTF Return Loss = 60dB; Cable Loss = 30dB; VSWR = 1; DTF VSWR = 1; Transmission = -90dB; Phase=-225

Default Units: dB

Example: To set trace bottom value in Return Loss to 40 dB:

```
:DISP:WIND:TRAC:Y:BOTT 40
```

:DISPlay:WINDow:TRACe:Y[:SCALe]:FULLscale

Cmd Description: Sets the top and bottom to their maximum values.

Example: To perform full-scale:

```
:DISP:WIND:TRAC:Y:FULL
```

:DISPlay:WINDow:TRACe:Y[:SCALe]:TOP <amplitude>{dB}
:DISPlay:WINDow:TRACe:Y[:SCALe]:TOP?

Cmd Description: Command sets the display top value.

Query Description: Query returns the display top value.

Param Type: Float

Default Value: Return Loss = 0dB; DTF Return Loss = 0dB; Cable Loss = 0dB; VSWR = 3; DTF VSWR = 3; Transmission = 10dB; Phase = 225

Default Units: dB

Example: To set trace top value in Return Loss to 10 dB:

```
:DISP:WIND:TRAC:Y:TOP 10
```

:FORMat[:READings][:DATA] ASCii|INTeger,32|REAL,32
:FORMat[:READings][:DATA]?

Cmd Description: Command sets the data format type. ASCii and REAL returns data in current unit. INTeger returns data with mdBm unit. This command applies to :TRACe[:DATA]? and :CALCulate:DATA?.

Query Description: Query returns the data format type as ASC or INT, 32 or REAL, 32.

Param Range: ASCii|INTeger,32|REAL,32

Example: To set data format type to integer:

```
:FORM INT,32
```

Notes: INTeger,32 values are signed 32-bit integers in little-endian byte order. This format returns the data in 4-byte blocks. REAL,32 values are 32-bit floating point numbers conforming to the IEEE 754 standard in little-endian byte order. This format returns the data in 4-byte binary format. The units are the current instrument units. Both INTeger and REAL formats return a definite block length. Each transfer begins with an ASCII header, such as #42204 for INTeger,32 and REAL,32. The first digit represents the number of following digits in the header.

:FORMat[:READings][:DATA]:MULTiplier 1|1M
:FORMat[:READings][:DATA]:MULTiplier?

Cmd Description: Command sets the data format multiplier factor.

Query Description: Query returns the data format multiplier factor. If '1M' is passed as argument, then query results for :TRACe[:DATA]? and :CALCulate:DATA? will specify readings multiplied by 10^6. If '1' is passed, then query results for :TRACe[:DATA]? and :CALCulate:DATA? will specify read.

Param Range: 1|1M

Default Value: 1M

Example: To set data format multiplier to 1:

```
:FORM:MULT 1
```

:INITiate#:DATA? FDATa|SDATa|FMEM|SMEM

Cmd Description: Command triggers a sweep and returns the data block of the specified trace at the end of sweep. If the system is in continuous mode, then this command is ignored.

Example: To trigger a sweep and wait for Formatted data to be returned:

```
:INIT1:DATA? FDAT
```

:INITiate:CONTinuous OFF|ON|0|1
:INITiate:CONTinuous?

Cmd Description: Command sets the sweep type between single sweep and continuous sweep.

Query Description: Query returns the sweep type, single or continuous. Query returns 1 for continuous sweep, and returns 0 for single sweep.

Param Range: OFF|ON|0|1

Default Value: ON|1

Example: To set the sweep type to single sweep:

```
:INIT:CONT OFF
```

:INITiate:HOLD OFF|ON|0|1
:INITiate:HOLD?

Cmd Description: Command sets the sweep state between run and hold. This command is ignored in external trigger mode.

Query Description: Query returns the sweep state, run or hold. Query returns 1 for hold mode, and returns 0 for run mode.

Param Range: OFF|ON|0|1

Default Value: OFF|0

Example: To set the sweep state to hold:

```
:INIT:HOLD ON
```

:INITiate[:IMMediate]

Cmd Description: Command triggers a single sweep if in single sweep mode. If the system is in continuous mode, then this command is ignored.

Example: To trigger a sweep:

```
:INIT
```

:SOURce:POWer LOW|HIGH
:SOURce:POWer?

Cmd Description: Command sets the source power level. Default is high.

Query Description: Query returns the source power level, LOW or HIGH.

Example: To set source power to low:

```
SOUR:POW LOW
```

:STATus:OPERation?

Query Description: Queries sweep status. This is used in combination with :INITiate[:IMMediate]. Sweep complete bit is the bit 8. Query returns 256 if sweep has completed. Otherwise, it returns 0.

Example: To query sweep status:

```
:STAT:OPER?
```

:TRACe[:DATA]? {1|2}

Query Description: Query returns raw trace data in real and imaginary format.

Param Type: Integer

Param Description: Optional Param: Trace number, default is 1.

Param Range: 1|2

Example: To get trace 1 data:

```
:TRAC?
```

Notes: The format of the block data that is returned can be specified by the command :FORMat[:READings][:DATA]. The response begins with an ASCII header that specifies the number of data bytes. It appears in the format #AX, where A is the number of digits in X, and X is the number of bytes that follow the header. Each data point is separated by a comma-delimiter. The returned numbers, which are independent of the Measurement Mode that is associated with the trace, are complex measurement data pairs (Real and Imaginary) for each point of the trace. A 517 point trace therefore has a total of 1034 values that get transferred.

[:SENSe]:APPLication:TST:RESult?

Query Description: Query returns self-test details. The self-test command (See [:SENSe]:APPLication:TST? [NORMal]) must be sent before getting the self-test details.

Example: To retrieve application self-test result:

```
:APPL:TST:RES?
```

Notes: The response begins with an ASCII header. The header specifies the number of following bytes. It appears in the format #AX, where A is the number of digits in X, and X is the number of bytes that follow the header. The first information of the result contains the overall self test string ("PASSED" or "FAILED") followed by a comma, and each self-test result separated by a comma. Each subset of the result is included in angled brackets.

[:SENSe]:APPLication:TST? {NORMal}

Query Description: Query returns 0 if self-test succeeds with no error. Otherwise, it returns 255.

Param Description: Optional Param: Self-test type, default is NORMal.

Param Range: NORMal

Example: To run an application self-test:

```
:APPL:TST?
```

[:SENSe]:CALibration:STATe?

Query Description: Query returns the state of calibrated measurements depending on the calibration collection type. 0 means no calibration was done. For calibration type RFP1 (OSL), return value is 1. For calibration type TRES (Transmission w/External Sensor), return value is 4. For calibration types 2PES (OSL + Transmission w/External Sensor), return value is 5.

Example: To query state after calibration is completed:

```
:CAL:STAT?
```

[:SENSe]:CORRection:COLLect:ABORt

Cmd Description: Aborts all steps of the RF calibration.

Example: To abort RF calibration:

```
:CORR:COLL:ABOR
```

[:SENSe]:CORRection:COLLect:INFO:STATus?

Query Description: Query returns 0 if calibration is not in progress, 1 if calibration is in progress, 2 if calibration was aborted, 3 if cal-coefficients are being calculated, or 4 if calibration has completed.

Example: To query calibration progress status:

```
:CORR:COLL:INFO:STAT?
```

[:SENSe]:CORRection:COLLect:INITialize

Cmd Description: Command to initialize an RF calibration with the current calibration type.

Example: To initialize RF calibration:

```
:CORR:COLL:INIT
```

[:SENSe]:CORRection:COLLect:LOAD

Cmd Description: Perform 'Load' step of calibration.

Example: To measure load:

```
:CORR:COLL:LOAD
```

[:SENSe]:CORRection:COLLect:OPEN

Cmd Description: Perform 'Open' step of calibration.

Example: To measure open:

```
:CORR:COLL:OPEN
```

[:SENSe]:CORRection:COLLect:SAVe

Cmd Description: Complete an RF Cal and perform coefficient calculation.

Example: To complete a RF calibration and perform coefficient calculation:

```
:CORR:COLL:SAV
```

[:SENSe]:CORRection:COLLect:SHORt

Cmd Description: Perform 'Short' step of calibration.

Example: To measure short:

```
:CORR:COLL:SHOR
```

[:SENSe]:CORRection:COLLect:STATus?
INITialize|OPEN|SHORt|LOAD|SAVE|ALL|THRU|ZERO

Query Description: Query returns 1 if a specified cal step is completed. Otherwise, it returns 0.

Param Range: INITialize|OPEN|SHORt|LOAD|SAVE|ALL|THRU|ZERO

Example: To query calibration step 'Open' status:

```
:CORR:COLL:STAT? OPEN
```

[:SENSe]:CORRection:COLLect:THRU

Cmd Description: Perform 'Thru' step of calibration.

Example: To measure Thru:

```
:CORR:COLL:THRU
```

[:SENSe]:CORRection:COLLect:TYPE RFP1|2PES|TRES
[:SENSe]:CORRection:COLLect:TYPE?

Cmd Description: Command sets calibration collection type.

Query Description: Query returns calibration collection type.

Param Range: RFP1|2PES|TRES, Classic Cable & Antenna Analyzer Mode only supports RFP1

Default Value: RFP1

Example: To set collection type to RFP1:

```
:CORR:COLL:TYPE RFP1
```

[:SENSe]:CORRection:COLLect:ZERO

Cmd Description: Perform 'Zero Sensor' step of calibration.

Example: To zero sensor:

```
:CORR:COLL:ZERO
```

[:SENSe]:CORRection:INSTacal:CALibrate

Cmd Description: Performs InstaCal.

Example: To perform a InstaCal:

```
:CORR:INST:CAL
```

[:SENSe]:CORRection:TYPE STANdard|FLEX
[:SENSe]:CORRection:TYPE?

Cmd Description: Command sets correction type.

Query Description: Query returns correction type. Query returns STAN|FLEX.

Param Range: STANdard|FLEX

Default Value: STANdard

Example: To set correction type to flex:

```
:CORR:TYPE FLEX
```

[:SENSe]:CORRection[:STATe] ON|OFF|1|0
[:SENSe]:CORRection[:STATe]?

Cmd Description: Command sets correction status.

Query Description: Query returns correction status. Query returns 1 if RF cal is on. Otherwise, it returns 0.

Param Range: ON|OFF|1|0

Default Value: OFF|0

Example: To turn calibration correction on:

```
:CORR:STAT ON
```

[:SENSe]:FREQuency:CABLe <cable-list-index>
[:SENSe]:FREQuency:CABLe?

Cmd Description: Command selects a cable with the index number from the sorted cable list.

Query Description: Query returns the index number of the selected cable based on the sorted cable list.

Param Type: Integer

Param Description: Index number from cable list

Default Value: 0

Example: To set the cable index to 10:

```
:FREQ:CABL 10
```

[:SENSe]:FREQuency:CABLe:CATalog:FULL?

Query Description: Query returns the entire cable list that is stored in memory.

Example: To get the cable list:

```
:FREQ:CABL:CAT:FULL?
```

[:SENSe]:FREQuency:CABLe:NAMe <"cable-name">
[:SENSe]:FREQuency:CABLe:NAMe?

Cmd Description: Command sets the cable name.

Query Description: Query returns the cable name.

Param Type: String

Param Description: Cable name

Default Value: NONE

Example: To set the cable to "AVA5-50 7/8":

```
:FREQ:CABL:NAM "AVA5-50 7/8"
```

[:SENSe]:FREQuency:STARt <frequency>{ Hz|kHz|MHz|GHz}
[:SENSe]:FREQuency:STARt?

Cmd Description: Command sets the start frequency.

Query Description: Query returns the start frequency in Hertz.

Param Type: Float

Param Description: Start frequency

Default Value: Model-dependent

Default Units: Hz

Example: To set the start frequency to 2 GHz:

```
:FREQ:STAR 2 GHZ
```

[:SENSe]:FREQuency:STOP <frequency>{ Hz|kHz|MHz|GHz}
[:SENSe]:FREQuency:STOP?

Cmd Description: Command sets the stop frequency.

Query Description: Query returns the stop frequency in Hertz.

Param Type: Float

Param Description: Stop frequency

Default Value: Model-dependent

Default Units: Hz

Example: To set the stop frequency to 2 GHz:

```
:FREQ:STOP 2 GHZ
```

[:SENSe]:RFON[:STATe] ON|OFF|1|0
[:SENSe]:RFON[:STATe]?

Cmd Description: Command sets the RF power output state when the sweep is in Hold mode.

Query Description: Query returns the RF power output state when the sweep is in Hold mode.

Param Range: ON|OFF|1|0

Default Value: ON|1

Example: To set RF power output state off:

```
:RFON OFF
```

[:SENSe]:ROSCillator[:SOURce]?

Query Description: Query returns reference oscillator setting. Query returns INT for internal or EXT for external.

Example: To query reference oscillator:

```
:ROSC?
```

[:SENSe]:SWEep:IFBW <frequency>{ Hz|kHz|MHz|GHz}
[:SENSe]:SWEep:IFBW?

Cmd Description: Command sets the IF bandwidth. The permissible values are 10Hz, 100Hz, 1kHz, 2kHz, and 100kHz.

Query Description: Query returns the IF bandwidth.

Param Type: Float

Param Description: Bandwidth frequency

Default Units: Hz

Example: To set IFBW to 1000Hz:

```
:SWE:IFBW 1 kHz
```

[:SENSe]:SWEep:RESolution 130|259|517|1033|2065
[:SENSe]:SWEep:RESolution?

Cmd Description: Command sets the data points.

Query Description: Query returns the data points as a number.

Param Range: 130|259|517|1033|2065

Default Value: 259

Example: To set data points to 1033:

```
:SWE:RES 1033
```

[:SENSe]:SWEep:RFIMmunity HIGH|LOW|1|0
[:SENSe]:SWEep:RFIMmunity?

Cmd Description: Command sets the RF immunity state.

Query Description: Query returns the RF immunity state, 1 for low immunity and 0 for high immunity.

Param Range: HIGH | LOW | 1 | 0

Default Value: LOW | 1

Example: To set RF immunity to low:

```
:SWE:RFIM LOW
```

[:SENSe]:SWEep:TYPE CONTinuous|SINGle|EXTernal

Cmd Description: Command sets the sweep trigger type to be Internal (Continuous or Single) or External.

Param Range: CONTinuous | SINGle | EXTernal

Default Value: CONTinuous

Example: To set the sweep trigger type to internal single sweep:

```
:SWE:TYPE SING
```

Notes: External Trigger setting is not available on all models.

[:SENSe]:TRACe:SELect 1|2
[:SENSe]:TRACe:SELect?

Cmd Description: Command sets the active trace.

Query Description: Query returns the active trace.

Param Range: 1 | 2

Default Value: 1

Example: To set active trace to trace 2:

```
:TRAC:SEL 2
```

Appendix A — Examples

A-1 C/C++

This example is run on the command line. It sends the *IDN? query to the instrument and prints the response to the console.

```
// IdnExample.cpp : Microsoft Visual Studio-Generated Example
//    Based on Example 2-1 in the NI-VISA User Manual
//    Usage : IdnExample "USB0::0x0B58::0xFFF9::xxxxxxxx_xxx_xx::INSTR"
//      where xxxxxxxx_xxx_xx is the USB Device ID of the
//      instrument.
//    Output : The string identity string returned from the
//      instrument.
//    VISA Header : visa.h (must be included)
//    VISA Libarary : visa32.lib  (must be linked with)

#include "stdafx.h"
#include "stdio.h"
#include "string.h"
#include "visa.h"

#define BUFFER_SIZE 255

int main(int argc, char* argv[])
{
  ViStatus status; /* For checking errors */
  ViSession defaultRM, instr; /* Communication channels */
  ViUInt32 retCount; /* Return count from string I/O */
  ViChar buffer[BUFFER_SIZE]; /* Buffer for string I/O */
  char tempDisplay[BUFFER_SIZE]; /* Display buffer for example */
  char *pAddress;


  /* Make sure we got our address. */
  if ( argc < 2 )
```

```
  {
    printf("Usage: IdnExample
\"USB0::0x0B58::0xFFF9::xxxxxxxx_xxx_xx::INSTR");
    printf("\t where xxxxxxxx_xxx_xx is the USB Device ID of your
instrument.\n");
    return -1;
  }

  /* Store the address. */
  pAddress = argv[1];

  /* Begin by initializing the system*/
  status = viOpenDefaultRM(&defaultRM);

  if (status < VI_SUCCESS)
  {
    /* Error Initializing VISA...exiting*/
    printf("Can't initialize VISA\n");
    return -1;
  }

  /* USB0::0x0B58::0xFFF9::xxxxxxxx_xxx_xx::INSTR*/
  /* NOTE: For simplicity, we will not show error checking*/
  /* TODO: Add error handling. */
  status = viOpen(defaultRM, pAddress, VI_NULL, VI_NULL, &instr);

  /* Set the timeout for message-based communication*/
  /* TODO: Add error handling. */
  status = viSetAttribute(instr, VI_ATTR_TMO_VALUE, 120000);

  /* Ask the device for identification */
  sprintf(buffer, "*IDN?\n");
  status = viWrite(instr, (unsigned char *)&buffer[0], 6, &retCount);
  status = viRead(instr, (unsigned char *)buffer, BUFFER_SIZE,
&retCount);

  /* TODO: Add code to process data. */
```

```
  strncpy(tempDisplay, buffer, retCount);
  tempDisplay[retCount] = 0;  /* Null-terminate display string. */
  printf("*IDN? Returned %d bytes: %s\n", retCount, tempDisplay);

  /* Close down the system */
  /* TODO: Add error handling. */
  status = viClose(instr);
  status = viClose(defaultRM);

  return 0;
}
```

A-2 Visual Basic

This function can be called in a Visual Basic program. It sends the *IDN? query to the instrument and returns the byte count and ASCII response string.

```
Rem This example is based on Example 2-1 from the NI-VISA User Manual.


Public Sub IdnMain(ByVal address As String, ByRef byteCount As String,
ByRef returnBytes As String)
    Const BUFFER_SIZE = 200
    Dim stat As ViStatus
    Dim dfltRM As ViSession
    Dim sesn As ViSession
    Dim retCount As Long
    Dim buffer As String * BUFFER_SIZE

    Rem ***Include visa32.dll as a reference in your project.***

    Rem Begin by initializing the system
    stat = viOpenDefaultRM(dfltRM)
    If (stat < VI_SUCCESS) Then
        Rem Error initializing VISA...exiting
        MsgBox "Can't initialize VISA"
        Exit Sub
    End If

    Rem Open communication with Device
    Rem NOTE: For simplicity, we will not show error checking
    Rem TODO: Add error handling.
    stat = viOpen(dfltRM, address, VI_NULL, VI_NULL, sesn)

    Rem Set the timeout for message-based communication
    Rem TODO: Add error handling.
    stat = viSetAttribute(sesn, VI_ATTR_TMO_VALUE, 120000)

    Rem Ask the device for identification
    Rem TODO: Add error handling.
```

```
    stat = viWrite(sesn, "*IDN?", 5, retCount)
    stat = viRead(sesn, buffer, BUFFER_SIZE, retCount)

    Rem TODO: Add code to process the data.
    byteCount = retCount
    returnBytes = Left(buffer, retCount)

    Rem Close down the system
    Rem TODO: Add error handling.
    stat = viClose(sesn)
    stat = viClose(dfltRM)
End Sub
```

A-3 Visual Basic

This function can be called in a Visual Basic program. It performs an RF Calibration in Cable & Antenna Analyzer mode. Communication with the instrument uses USB protocol.

```
Public Sub OnePortCalibrationInCAAMode()


    Const MAX_CNT = 200
    Dim stat As Variant
    Dim dfltRM As Variant
    Dim sesn As Variant
    Dim retCount As Long
    Dim Buffer As String * MAX_CNT
    Dim Response As String * VI_FIND_BUFLEN
    Dim sInputString As String

    Rem Begin by initializing the system
    stat = viOpenDefaultRM(dfltRM)

    If (stat < VI_SUCCESS) Then
        Rem Error initializing VISA...exiting
        Exit Sub
    End If

    Rem Open communication with USB Protocol
    Rem NOTE: For simplicity, we will not show error checking
    Rem 0x0B5B::0xFF60::32850021_76227-3_102 = Vendor id::Product
id::dut usb id
    stat = viOpen(dfltRM,
"USB0::0x0B5B::0xFF60::32850021_76227-3_102::INSTR", VI_NULL, VI_NULL,
sesn)

    Rem Set some visa attributes
    stat = viSetAttribute(sesn, VI_ATTR_TMO_VALUE, 90000)
    stat = viSetAttribute(sesn, VI_ATTR_SEND_END_EN, VI_TRUE)
    stat = viSetAttribute(sesn, VI_ATTR_SUPPRESS_END_EN, VI_FALSE)
    stat = viClear(sesn)

    'Switch to Cable-Antenna Analyzer Mode
```

```
sInputString = ":INST:NSEL 2"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
Do
    Sleep (200)
    sInputString = ":INST:NSEL?"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)
Loop Until Val(Buffer) = 2

'System preset
sInputString = ":SYSTEM:PRESET"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

'Wait for previous operation to be completed
sInputString = "*OPC?"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
Buffer = ""
stat = viRead(sesn, Buffer, MAX_CNT, retCount)

'Set start frequency
sInputString = ":SENSe:FREQuency:STARt 2 MHz"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

'Set stop frequency
sInputString = "SENSe:FREQuency:STOP 4 GHz"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

'Initiate One-port Calibration
sInputString = "SENSe:CORRection:COLLect:TYPE RFP1"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

'Wait for previous operation to be completed
sInputString = "*OPC?"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
Buffer = ""
stat = viRead(sesn, Buffer, MAX_CNT, retCount)
```

```
sInputString = "SENSe:CORRection:COLLect:INITialize"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

'Wait for previous operation to be completed
sInputString = "*OPC?"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
Buffer = ""
stat = viRead(sesn, Buffer, MAX_CNT, retCount)

'measure open
MsgBox "Connect open at port 1"

sInputString = ":SENSe:CORRection:COLLect:OPEN"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
Do
    Sleep (200)
    'wait open measurement to complete and returns 1
    sInputString = ":SENSe:CORRection:COLLect:STATus? OPEN"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)
Loop Until Val(Buffer) = 1

'measure short
MsgBox "Connect short at port 1"

sInputString = ":SENSe:CORRection:COLLect:SHORt"
stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
Do
    Sleep (200)
    'wait short measurement to complete and returns 1
    sInputString = ":SENSe:CORRection:COLLect:STATus? SHORt"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)
Loop Until Val(Buffer) = 1

'measure load
```

```
    MsgBox "Connect load at port 1"

    sInputString = ":SENSe:CORRection:COLLect:LOAD"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Do
        Sleep (200)
        'wait load measurement to complete and returns 1
        sInputString = ":SENSe:CORRection:COLLect:STATus? LOAD"
        stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
        stat = viRead(sesn, Buffer, MAX_CNT, retCount)
    Loop Until Val(Buffer) = 1

    'Save and apply calibration
    sInputString = ":SENS:CORR:COLL:SAV"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

    'Wait for previous operation to be completed
    sInputString = "*OPC?"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Buffer = ""
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)

    'read back the the cal type (i.e. Buffer = 1, RFP1 or one-port
calibration)
    sInputString = ":SENS:CAL:STAT?"
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Buffer = ""
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)

    Rem Close down the system
    stat = viClose(sesn)
    stat = viClose(dfltRM)
End Sub
```

A-4 Visual Basic

This function can be called in a Visual Basic program. It demonstrates connection and setting parameters in the instrument while using Ethernet Socket protocol.

```
Public Sub CommunicationWithTCPIPSocket()

    Const MAX_CNT = 200
    Dim stat As Variant
    Dim dfltRM As Variant
    Dim sesn As Variant
    Dim retCount As Long
    Dim Buffer As String * MAX_CNT
    Dim Response As String * VI_FIND_BUFLEN
    Dim sInputString As String
    Dim ipAddress As String
    Dim Port As String

    Rem Begin by initializing the system
    stat = viOpenDefaultRM(dfltRM)

    If (stat < VI_SUCCESS) Then
        Rem Error initializing VISA...exiting
        Exit Sub
    End If

    Rem Open communication with Ethernet Socket Protocol
    Rem before open an new Ethernet session make sure session was closed
    Rem NOTE: For simplicity, we will not show error checking

    'address and port

    'this sample address
    ipAddress = "172.26.202.117"
    'For S331L port will be 9001
    Port = "9001"
```

```
    stat = viOpen(dfltRM, "TCPIP0::" & ipAddress & "::" & Port &
"::SOCKET", VI_NULL, VI_NULL, sesn)

    Rem Set some visa attributes

    Rem recommandation timeout >= 90 sec
    stat = viSetAttribute(sesn, VI_ATTR_TMO_VALUE, 90000)
    stat = viSetAttribute(sesn, VI_ATTR_SEND_END_EN, VI_TRUE)
    Rem VI_ATTR_SUPPRESS_END_EN has to set to False during Ethernet
Socket communication
    stat = viSetAttribute(sesn, VI_ATTR_SUPPRESS_END_EN, VI_FALSE)
    stat = viClear(sesn)

    Rem NOTE:
    Rem All commands (SCPI) must be send with linefeed
    Rem during Ethernet Socket communication
    Rem i.e. "vbLf" is in Visual Basic environment constant

    'read back the strat frequency
    sInputString = "*IDN?" & vbLf
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Buffer = ""
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)

    'System preset
    sInputString = ":SYSTEM:PRESET" & vbLf
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

    'Wait for previous operation to be completed
    sInputString = "*OPC?" & vbLf
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Buffer = ""
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)

    'Set start frequency
    sInputString = ":SENSe:FREQuency:STARt 1 GHz" & vbLf
```

```
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

    'read back the strat frequency
    sInputString = ":SENSe:FREQuency:STARt?" & vbLf
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Buffer = ""
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)

    'Set stop frequency
    sInputString = "SENSe:FREQuency:STOP 4 GHz" & vbLf
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)

    'read back the stop frequency
    sInputString = ":SENSe:FREQuency:STOP?" & vbLf
    stat = viWrite(sesn, sInputString, Len(sInputString), retCount)
    Buffer = ""
    stat = viRead(sesn, Buffer, MAX_CNT, retCount)


    Rem Close down the system
    stat = viClose(sesn)
    stat = viClose(dfltRM)

End Sub
```

A-5 LabVIEW™

This example shows how to read the trace data from the instrument in 32-bit integer format. The output is an array of data point magnitudes. Figure 1 shows the data capture and conversion to 32-bit integers in the format used by LabVIEW. Figure 2 shows the details of the conversion.

Note	Your instrument must first be defined to the VISA resource manager using NI-MAX. The VISA resource for your instrument serves as the VISA resource input to the vi.

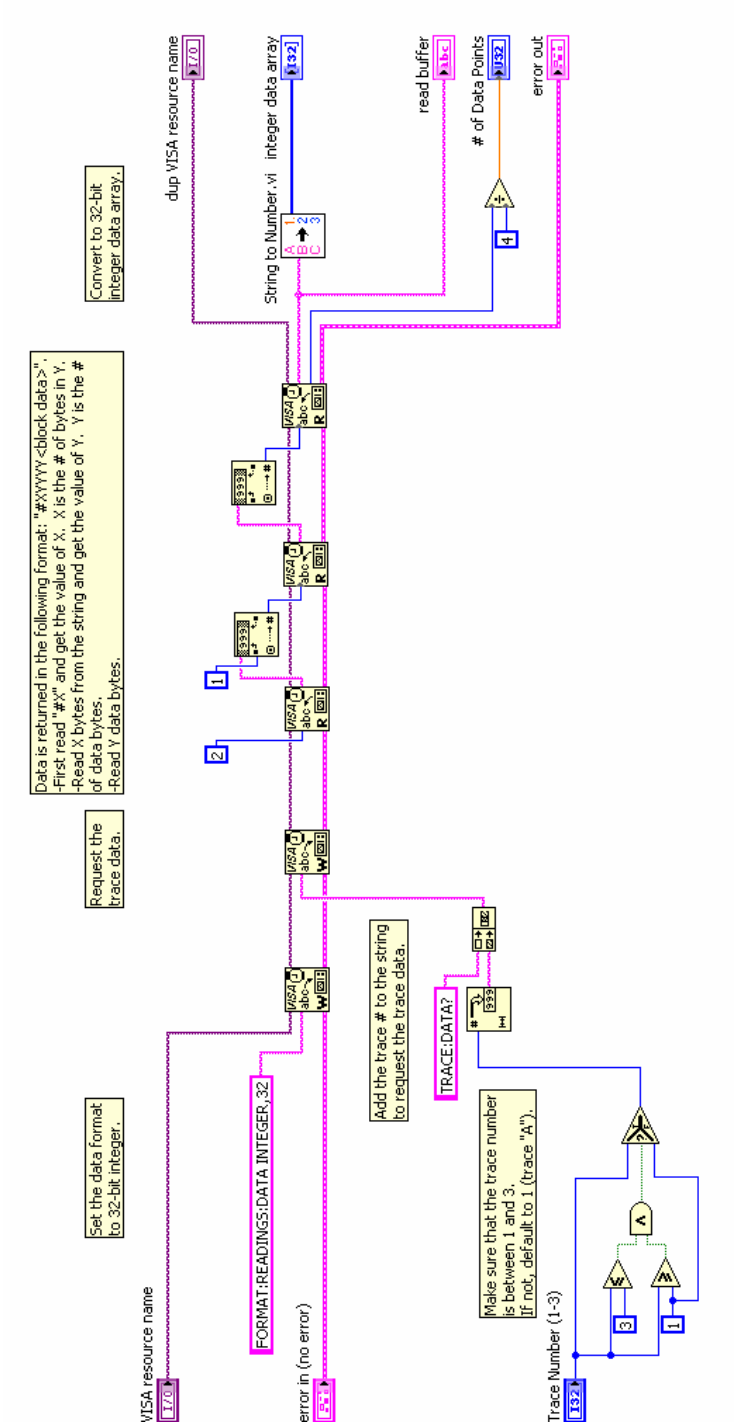


Figure A-1. Data Capture

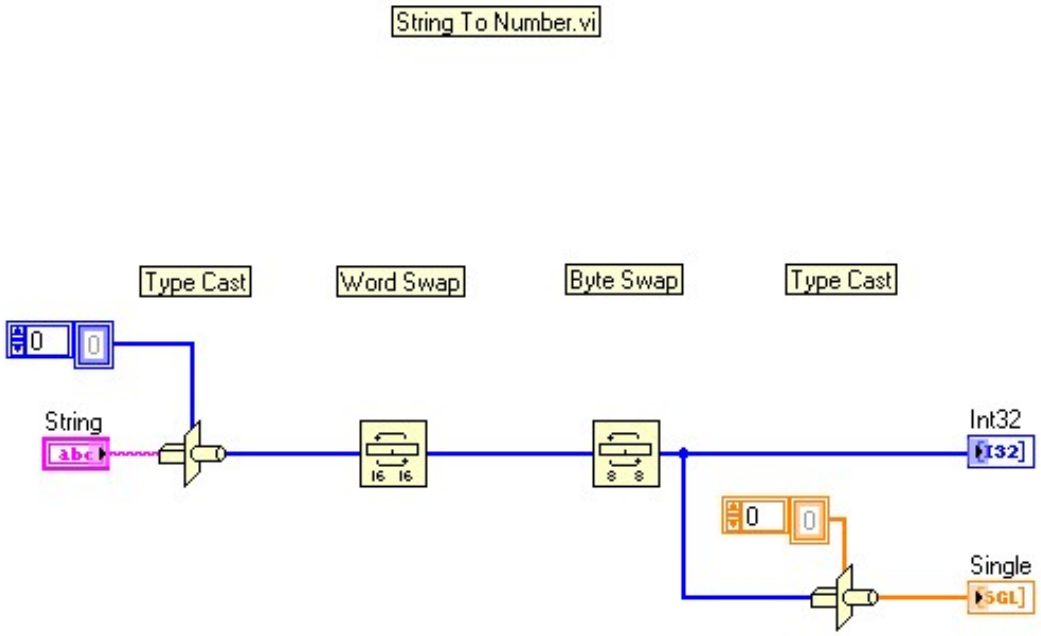


Figure A-2. Data Conversion

Appendix B — List of Commands by Mode

Chapter 1—General Information

Chapter 2—Programming with SCPI

Chapter 3—All Mode Commands

Chapter 4—Cable & Antenna Analyzer Mode Commands

Appendix C — List of Commands, Alphabetical

All SCPI Commands in Alphabetic List

Anritsu

10580-00322

E

Anritsu utilizes recycled paper and environmentally conscious inks and toner.

Anritsu Company
490 Jarvis Drive
Morgan Hill, CA 95037-2809
USA
http://www.anritsu.com