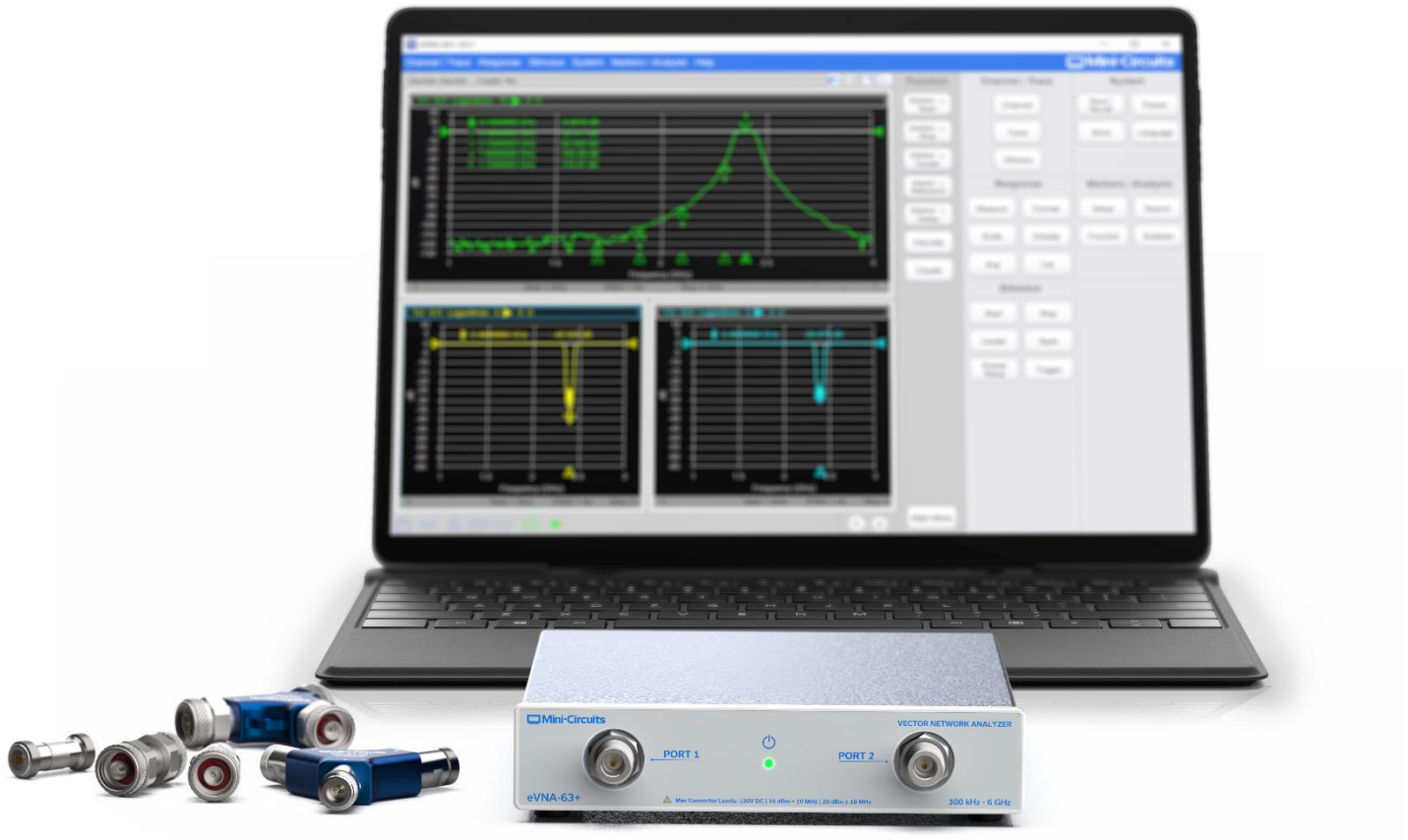
Programming Manual

VECTOR NETWORK ANALYZER

eVNA-63+



testsolutions@minicircuits.com | www.minicircuits.com

Contents

1. Overview

Mini-Circuits' eVNA-63+ is a high performance, software-controlled vector network analyzer (VNA). By moving the complex data processing and calculation required of vector network measurements out of the instrument and into an advanced software package, Mini-Circuits is able to offer a fully featured but cost effective VNA for every test bench.

The product ships with Mini-Circuits' eVNA Studio user interface (UI) software, providing a powerful GUI which will feel familiar to any engineer with experience of VNA measurements. This programming manual is intended for customers who wish to go further and implement their own control program using the eVNA Application Programming Interface (API), which allows automation of VNA calibrations, measurements, trace displays and data exports.

1.1. Concept & Control Options

Mini-Circuits' eVNA concept is a system formed from 3 elements, all of which are required for VNA functionality:

1. Host PC with Microsoft Windows
2. eVNA-63+ hardware connected by USB to the host PC
3. eVNA Studio software on the host PC

The eVNA system can be controlled using the eVNA Studio UI on the host PC. Custom automation programs based on the eVNA API can be implemented on the host PC or a remote PC via a TCP / IP connection.

Any references in this programming manual to eVNA or VNA are intended to refer to the complete system.

1.2. eVNA User Guide

For complete instructions on operation of the eVNA, including use of the eVNA Studio UI, please refer to:

www.minicircuits.com/WebStore/Vector-Network_Analyzer.html

1.3. Support Contacts

We are here to support you every step of the way. For technical support and assistance, please contact us at the email address below or refer to our website for your local support:

testsolutions@minicircuits.com

www.minicircuits.com/contact/worldwide_tech_support.html

2. Software Installation

2.1. Software Downloads & Resources

The complete eVNA Studio software package is available for download from:

www.minicircuits.com/softwaredownload/evna.html

2.2. System Requirements

The basic requirements for installation of the eVNA software package and API on the host PC are:

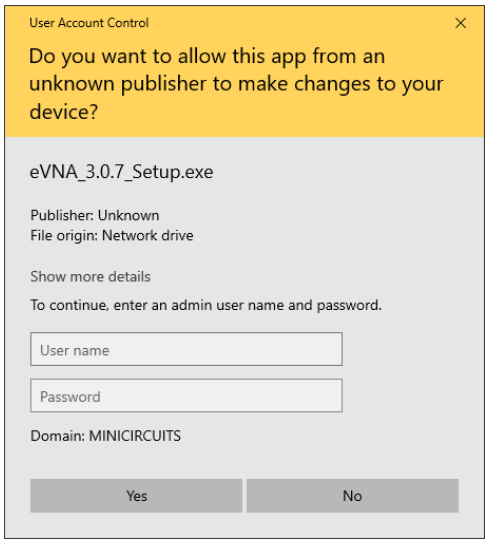
- Microsoft Windows 7 or 10 (64-bit)
- Intel i3 processor or equivalent
- 8 GB RAM
- USB 2.0 or later
- Ethernet connection (for remote access to the API over a TCP / IP network)

2.3. eVNA Studio Installation (GUI & API Package)

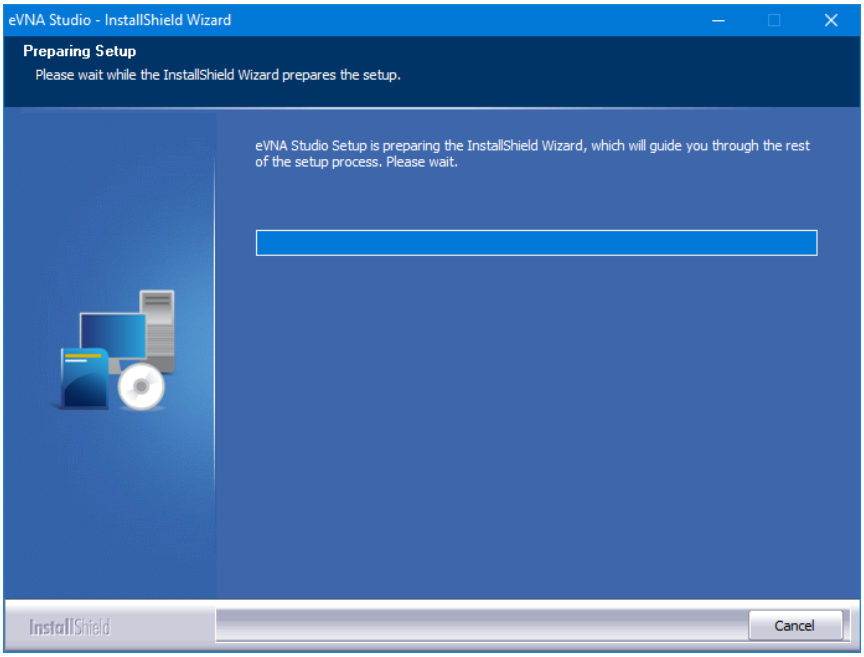
1. Download the eVNA View software package from:

   https://www.minicircuits.com/softwaredownload/eVNA_Setup.zip

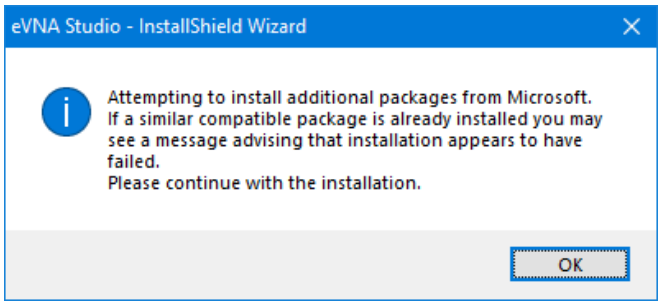
2. Run the eVNA_x.x.x_Setup.exe installation program.
3. If prompted by Windows User Account Control, enter the username and password for an account with permission to install software on the PC.



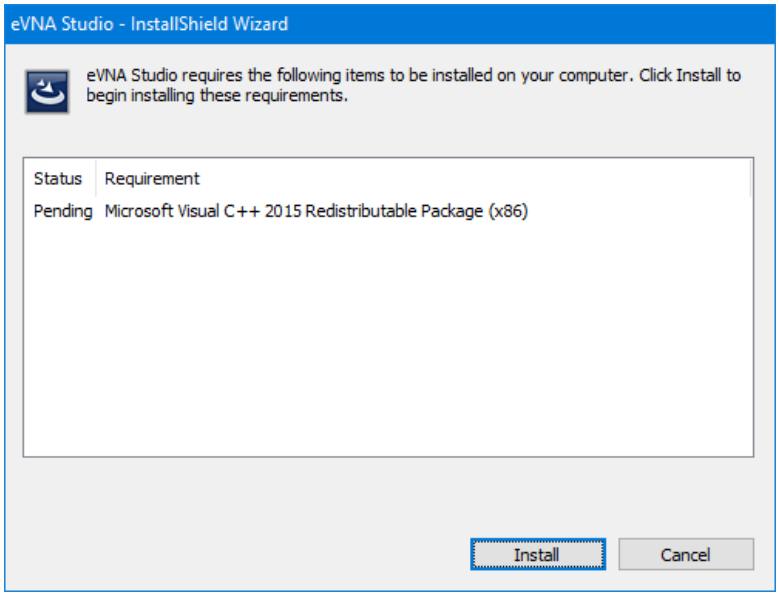
4. The InstallShield Wizard will load, followed by a Windows Security prompt asking for permission to install the software.



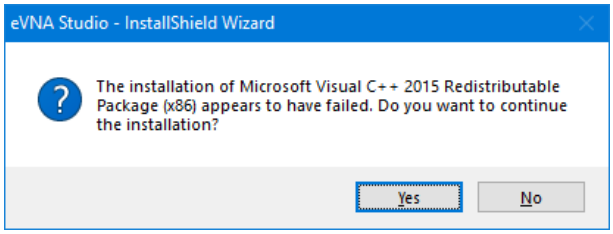
5. Click OK on the prompt advising that installation of additional packages from Microsoft will be attempted.



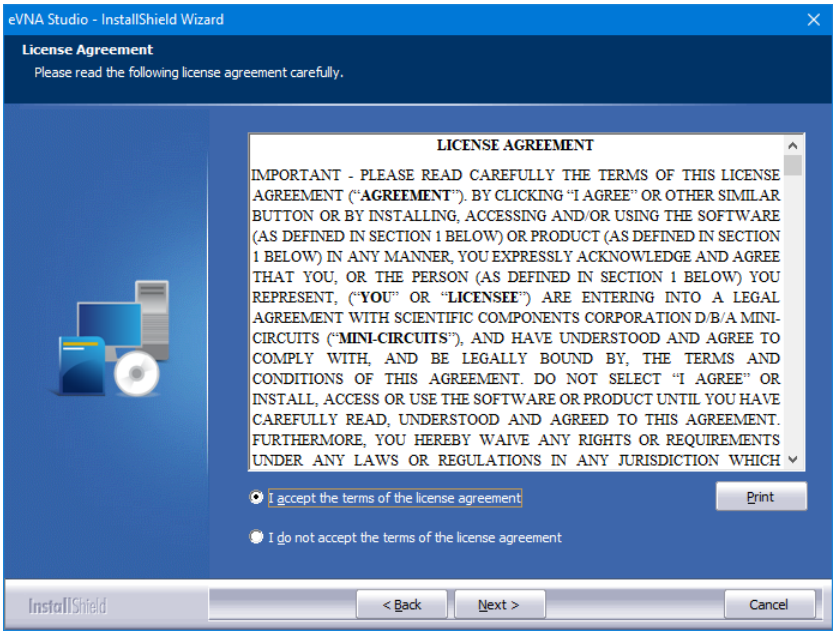
6. Click Install to confirm the installation of "Microsoft Visual C++ 2015 Redistributable Package".



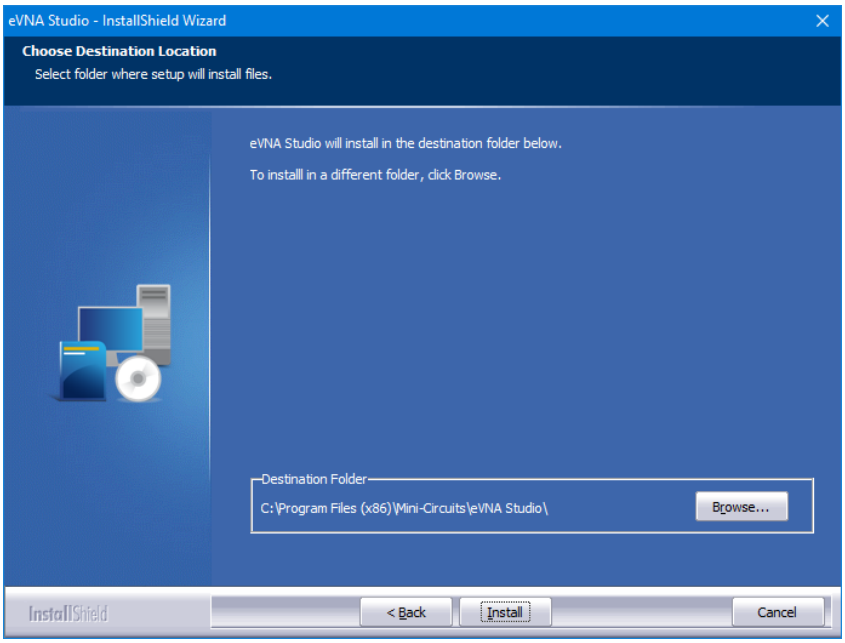
7. The installation of "Microsoft Visual C++ 2015 Redistributable Package" will fail if a comparable version is already installed. Click Yes to continue with the installation anyway if the warning message appears.



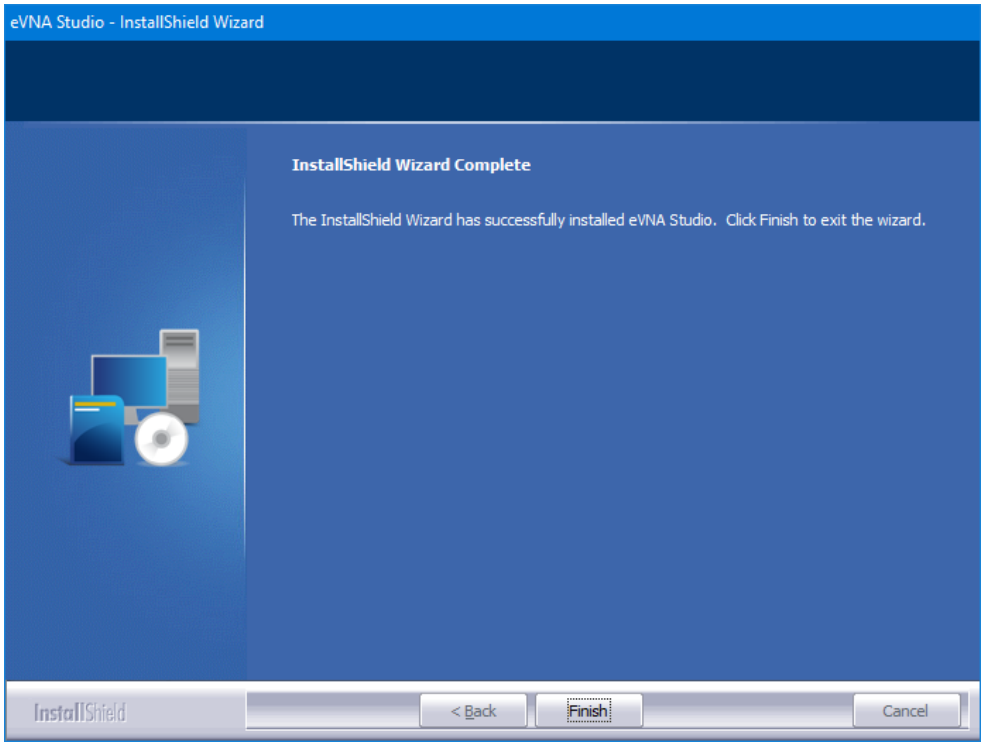
8. The InstallShield Wizard for the eVNA View software package will load, click Next to continue.
9. Accept the terms of the license agreement and click Next to continue.



10. Click Install to accept the default location (you may first click Browse to select an alternative installation folder if preferred).



11. The final screen will confirm installation of the software, click Finish to finalize.



In the event of any issues with installation, please contact testsolutions@minicircuits.com for support.

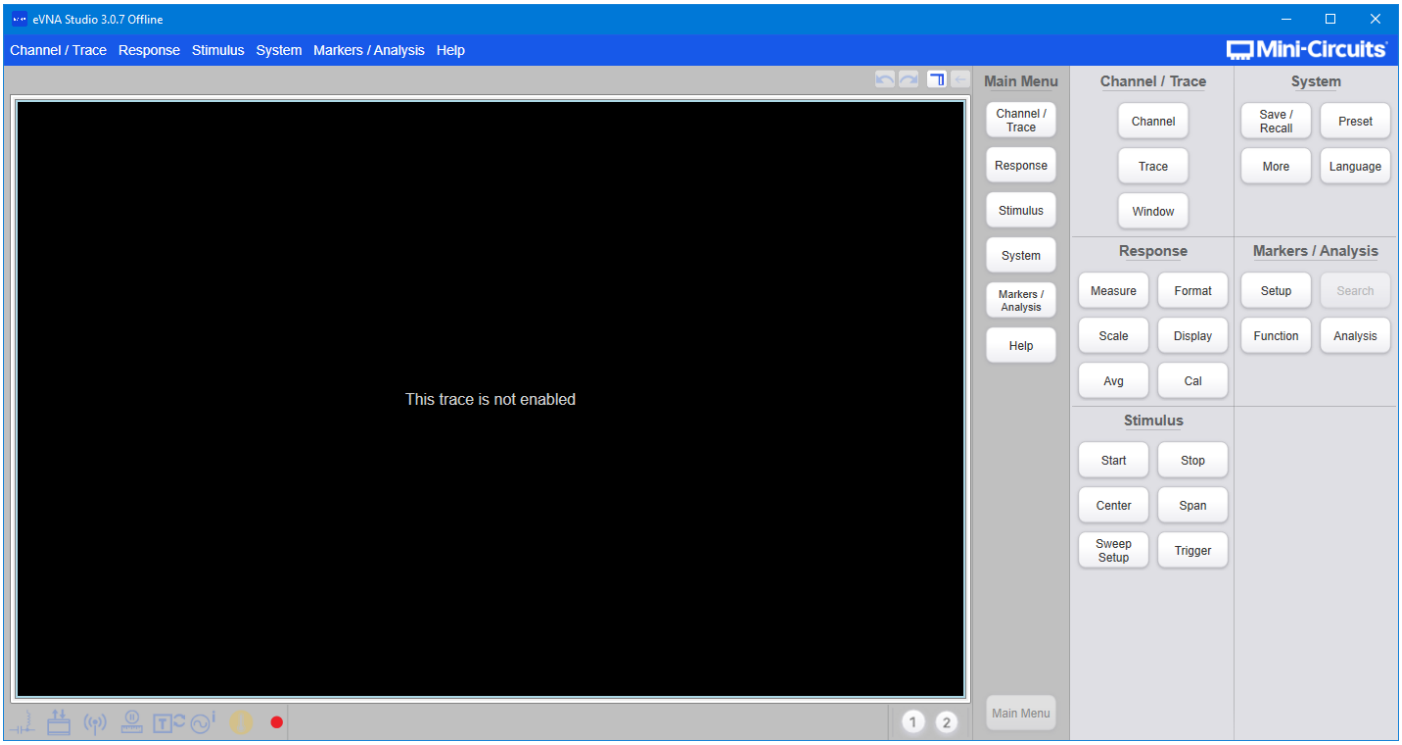
3. API Configuration

3.1. Launch eVNA Studio

The eVNA Studio software must be running / open on the host PC in order to control the eVNA using either the UI or API.

1. Double-click the eVNA Studio icon on the desktop or Windows start menu

2. The GUI will open
3. All further steps, including identifying and connecting the eVNA-63+ instrument connected by USB, can be carried out using the UI or the SCPI commands detailed later in this manual
4. The eVNA Studio must be left open on the host PC in order for the API to operate, even if controlling from a remote PC



3.2. Socket Communication

Communication with the eVNA API is implemented by establishing a TCP / IP raw socket connection using the IP address of the host PC and port 5026. Where the automation program is implemented on the host PC, the localhost IP address can be used (127.0.0.1).

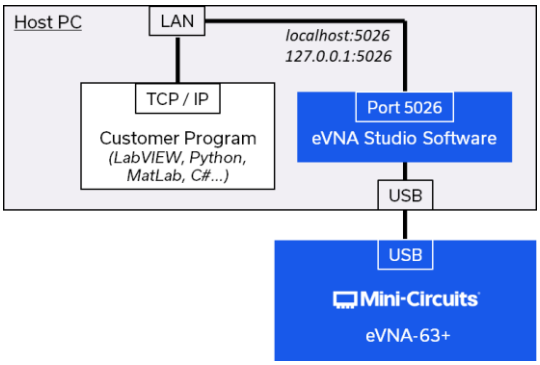
The SCPI commands / queries summarized below should be terminated with the `\n` line-feed character.

Note: Some standard TCP / IP clients terminate commands with the `\r\n` (carriage-return & linefeed) characters. The eVNA may stop responding to subsequent commands when `\r` is the first indication that a command has ended. When this behavior is seen, adding a space character at the end of each SCPI command is sufficient to ensure proper operation.

3.2.1. USING THE API ON THE HOST PC

The host PC is connected by USB to the eVNA-63+ hardware. A custom program implemented on the host PC needs to establish a socket connection to the API using the "localhost" IP address (127.0.0.1) and port 5026.

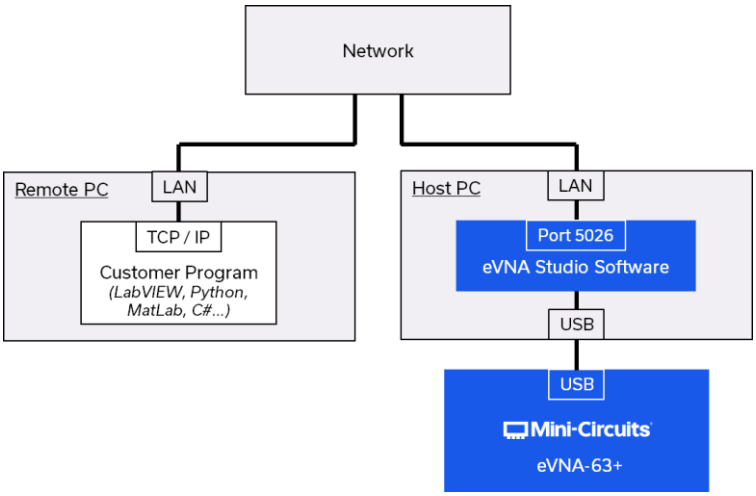
The eVNA Studio software must be installed and open on the host PC in order to use the API.



3.2.2. USING THE API FROM A REMOTE PC

The host PC is connected by USB to the eVNA-63+ hardware. A custom program can be implemented on a remote PC, connected via a TCP / IP network to the host PC. The program needs to establish a socket connection to the API using the IP address of the host PC and port 5026.

The eVNA Studio software must be installed and open on the **host** PC in order to use the API.



3.2.3. CONNECTION EXAMPLE

The example script below demonstrates establishing a connection and sending SCPI commands using Python.

```
import sys
import socket
import time

#evna_ip = socket.gethostbyname('HOSTNAME')     # Obtain the IP of the host PC by hostname
evna_ip = 'localhost'   # IP address of the host PC (or localhost / 127.0.0.1)
evna_port = 5026        # eVNA Studio requires port 5026

# Function to send SCPI command (no response expected)
def SCPI_Command(scpi):
    s.send(str.encode(scpi + '\n'))             # Send (add new line and encode)
    print (scpi)

# Function to send SCPI query and get response
def SCPI_Query(scpi):
    s.send(str.encode(scpi + '\n'))             # Send (add new line and encode)

    try:
        response_list = []                              # List to collect the response in mutiple chunks
        while True:                                     # Loop to collect each chunk of the response
            response_chunk = s.recv(1024)               # Get the response (up to max buffer size)
            response_list.append(str(response_chunk, 'utf-8'))  # Append it to the list
            if '\n' in str(response_chunk, 'utf-8'):    # Keep looping until new line character received
                break

        evna_response = ''.join(response_list)          # Join the list of responses into a sinle string

    except Exception as e:
        evna_response = 'Query failed: ' + str(e)  # Check error

    print (scpi, '==>', evna_response)
    return evna_response

# ------------------------------------------------
# Connect to the eVNA Studio GUI / API on host PC
# ------------------------------------------------

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(3)     # Set timeout in seconds
# Note if using *OPC *OPC? *WAI for synchonisation of operations:
# - Ensure the API socket timeout is set longer than the operation time, to avoid timeout errors.
try:
    s.connect((evna_ip, evna_port))         # Connect to eVNA Studio
    print ('Connected to eVNA Studio on host PC:', evna_ip, str(evna_port))
except Exception as e:                      # Connection failed
    print('Could not connect:', evna_ip, evna_port, str(e))
    s.close()
    sys.exit(0)

# ------------------------------------------------
# Find & connect an eVNA-63+ instrument
# ------------------------------------------------

evna_list = SCPI_Query('SYSTEM:DISCOVER?')      # Search for available eVNA instruments, connected to host PC
evna_name = evna_list.split(',')                # Comma separated list of eVNA names
SCPI_Command('SYSTEM:CONNECT ' + evna_name[0])  # Connect the first eVNA found

evna_id = SCPI_Query('*IDN?')                   # Identify the connected eVNA
evna_id_parts = evna_id.split(',')              # Comma separated list of identification parameters
if evna_id_parts[0] != 'EVNA63':                # Confirm a real instrument connected (rather than simulator)
    print("eVNA instrument connection failed")
    s.close()
    sys.exit(0)

SCPI_Command('SYSTEM:PRESET')                   # Reset eVNA configuration to the preset state
```

3.3. VISA Communication (Limited)

The eVNA does not include native support for VISA (Virtual Instrument Software Architecture) so it will not be discovered automatically in a VISA system. The eVNA can still be controlled using VISA libraries by manually configuring it as a VISA LAN instrument within the management software, included with the VISA package.

VISA is a commonly used API for communication with test and measurement equipment. It has become an industry standard maintained by the IVI foundation (http://ivifoundation.org), as a vendor agnostic method to support communication with a wide range of instrumentation from a wide range of manufacturers.

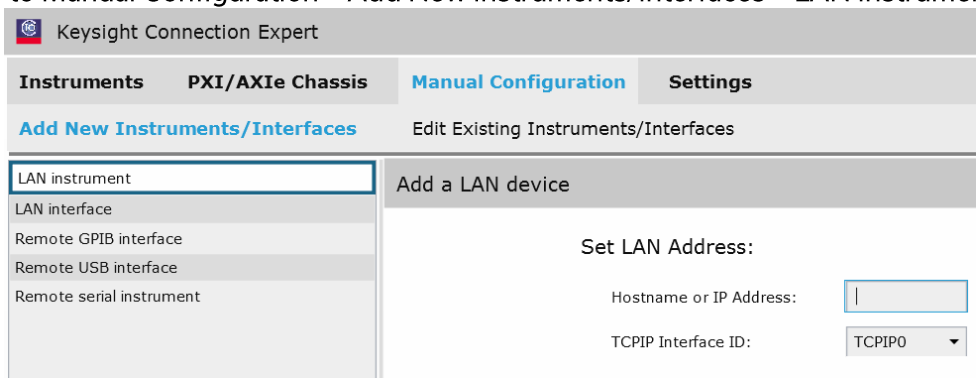
Several VISA implementations are available at no cost. Two of the most commonly used implementations are:

- Keysight IO Libraries Suite (www.keysight.com)
- NI-VISA (www.ni.com)

3.3.1. CREATING A LAN INSTRUMENT WITH KEYSIGHT CONNECTION EXPERT

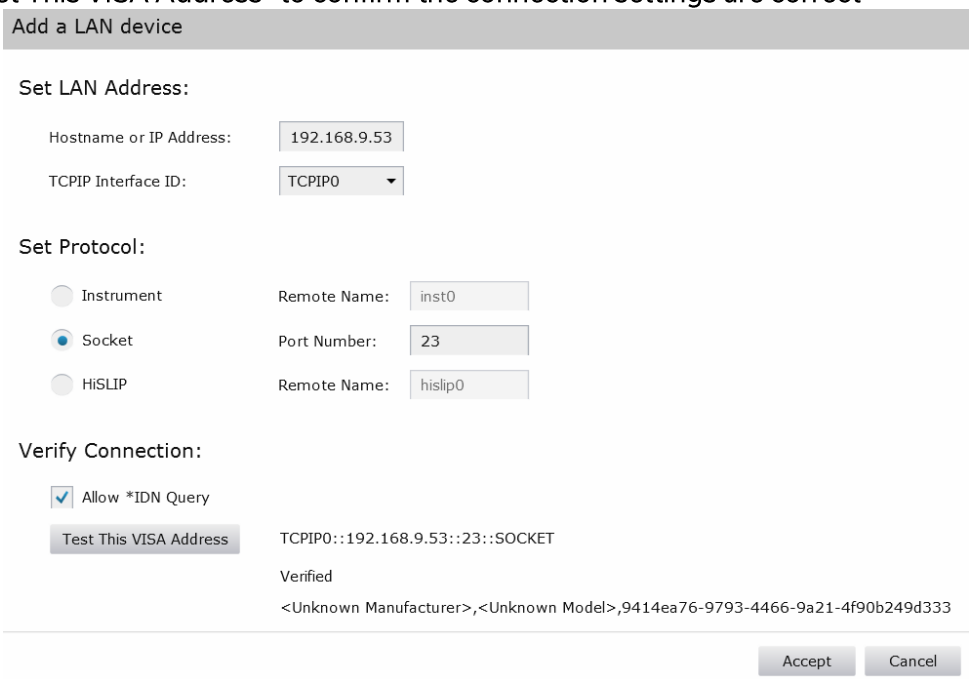
Since eVNA-63+ does not support VISA as standard it is necessary to configure the device as a new LAN instrument using the following steps:

1. Launch Keysight Connection Expert
2. Navigate to Manual Configuration > Add New Instruments/Interfaces > LAN Instrument

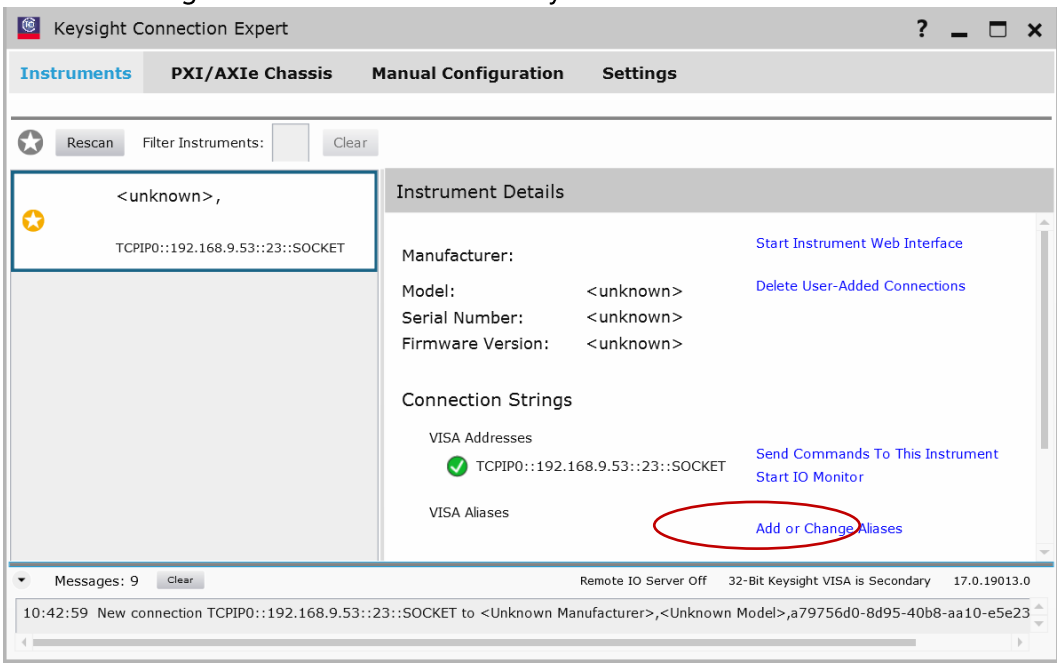


3. Enter the IP address of the host PC (connected by USB to the eVNA-63+ hardware) in the "Hostname or IP Address" field. For VISA control running on the host PC, use the localhost IP address (127.0.0.1).
4. "TCPIP Interface ID" should usually be left as the default, this is the ID for the LAN itself
5. In the "Set Protocol" section, select "Socket" and enter port number "5026"

6. Click "Test This VISA Address" to confirm the connection settings are correct



7. Click Accept to save the configuration and Connection Expert will return to the home screen where the new instrument will be listed along with any other known VISA devices.
8. The eVNA-63+ will be listed as "Unknown" but will be identifiable by the IP address
9. To add an "alias" in order to make the device more easily identifiable, click on the device and then select "Add or Change Aliases" from the summary screen

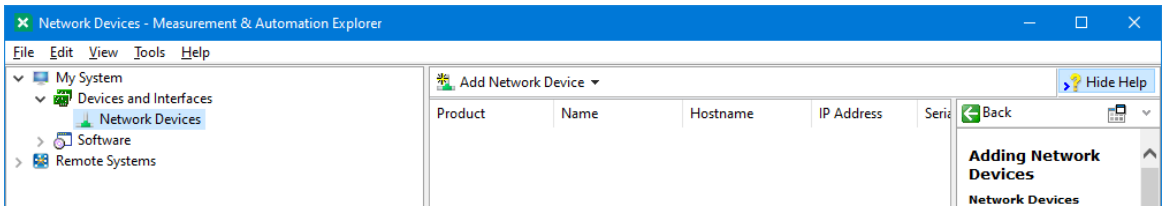


10. Enter the chosen Alias Name and make sure the correct VISA Address string is selected if there are multiple VISA instruments defined
11. The device is now configured and ready to use with a VISA address that takes the form "[LAN_Interface]::[IP_Address]::[Port]::SOCKET"

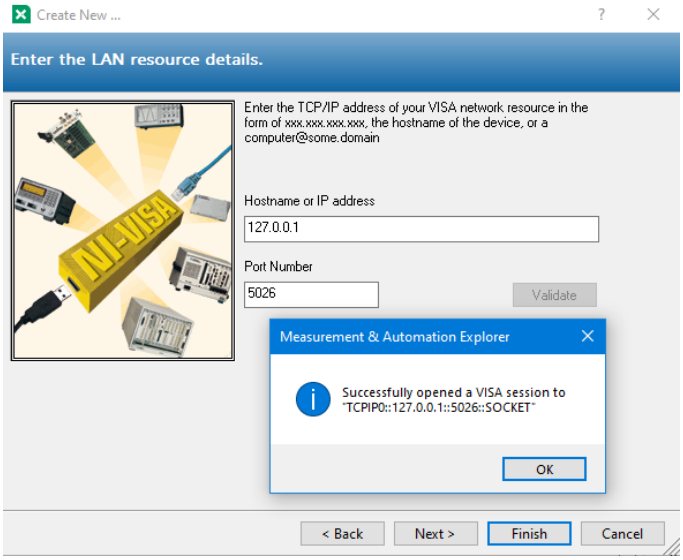
3.3.2. CREATING A LAN INSTRUMENT WITH NI MAX

Since eVNA-63+ does not support VISA as standard it is necessary to configure the device as a new LAN instrument using the following steps:

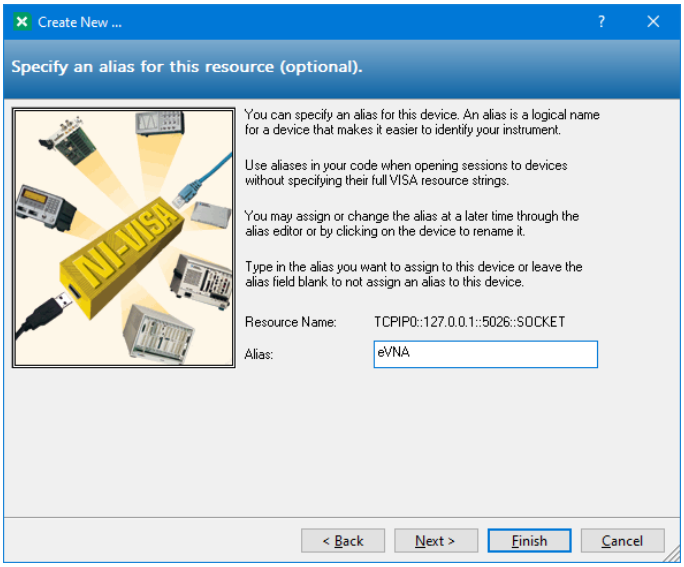
1. Launch NI MAX (Measurement & Automation Explorer)
2. In the navigation column on the left of the screen, expand "My System" and then "Devices and Interfaces"
3. Rick-click on "Network Devices" and select "Create New VISA TCP/IP Resource"



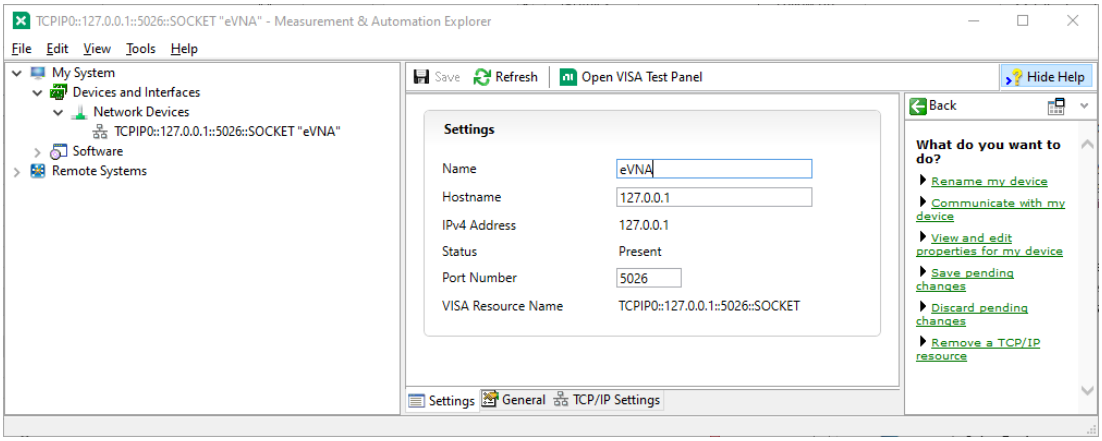
4. Select "Manual Entry of Raw Socket" from the pop-up screen and click Next
5. Enter the IP address of the host PC (connected by USB to the eVNA-63+ hardware) and port number "5026". For VISA control running on the host PC, you can also use localhost or the related IP address (127.0.0.1).
6. Click **Validate** to test the connection



7. Click **Next** and enter a convenient name in the Alias field



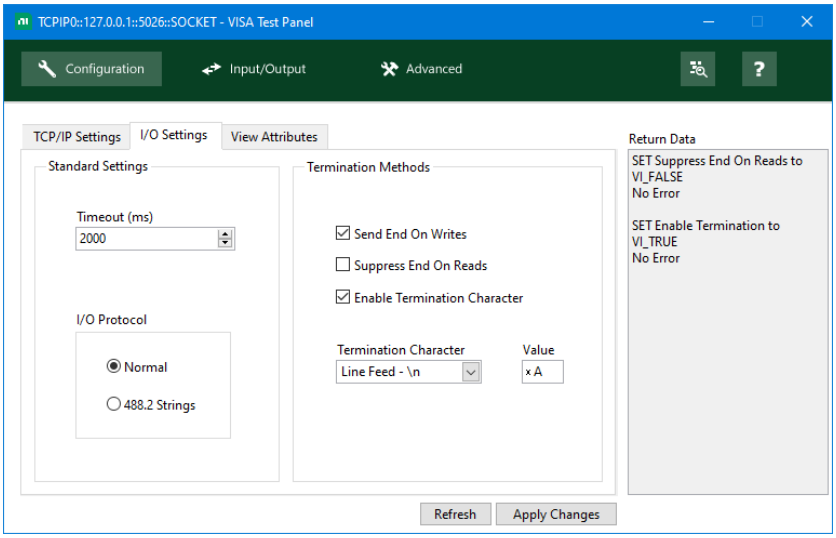
8. Click Finish to return to the home screen with the new device now listed under the "Network Devices" heading



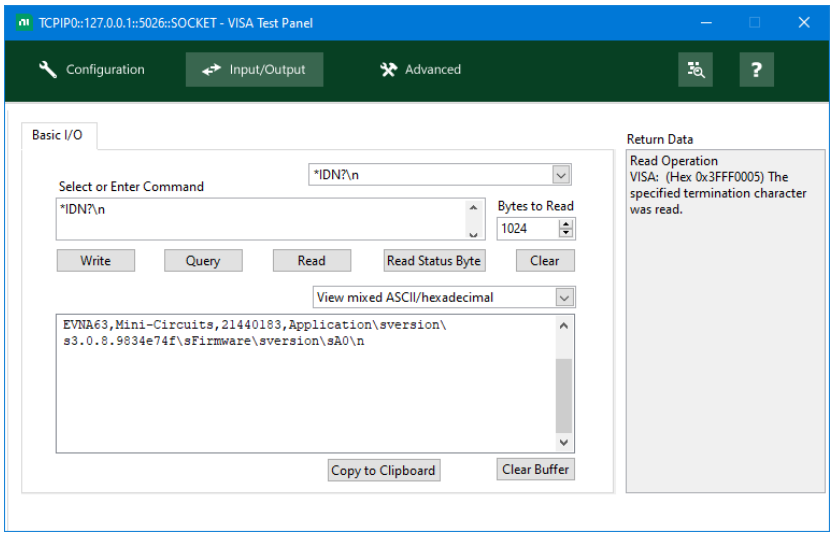
9. The device is now configured and ready to use with a VISA address that takes the form "[LAN_Interface]::[IP_Address]::[Port]::SOCKET"

The VISA Test Panel within NI MAX can be used to test sending SCPI commands to the eVNA:

1. Select the device defined above in the NI MAX main screen and click **Open VISA Test Panel**
2. Go to Configuration and I/O Settings:
   a. Uncheck "Suppress End on Reads"
   b. Check "Enable Termination Character"
   c. Set the termination character as **\n**



3. Queries can be sent to the eVNA in the Input/Output tab, using \n as the termination character



3.4. Control of Multiple VNAs

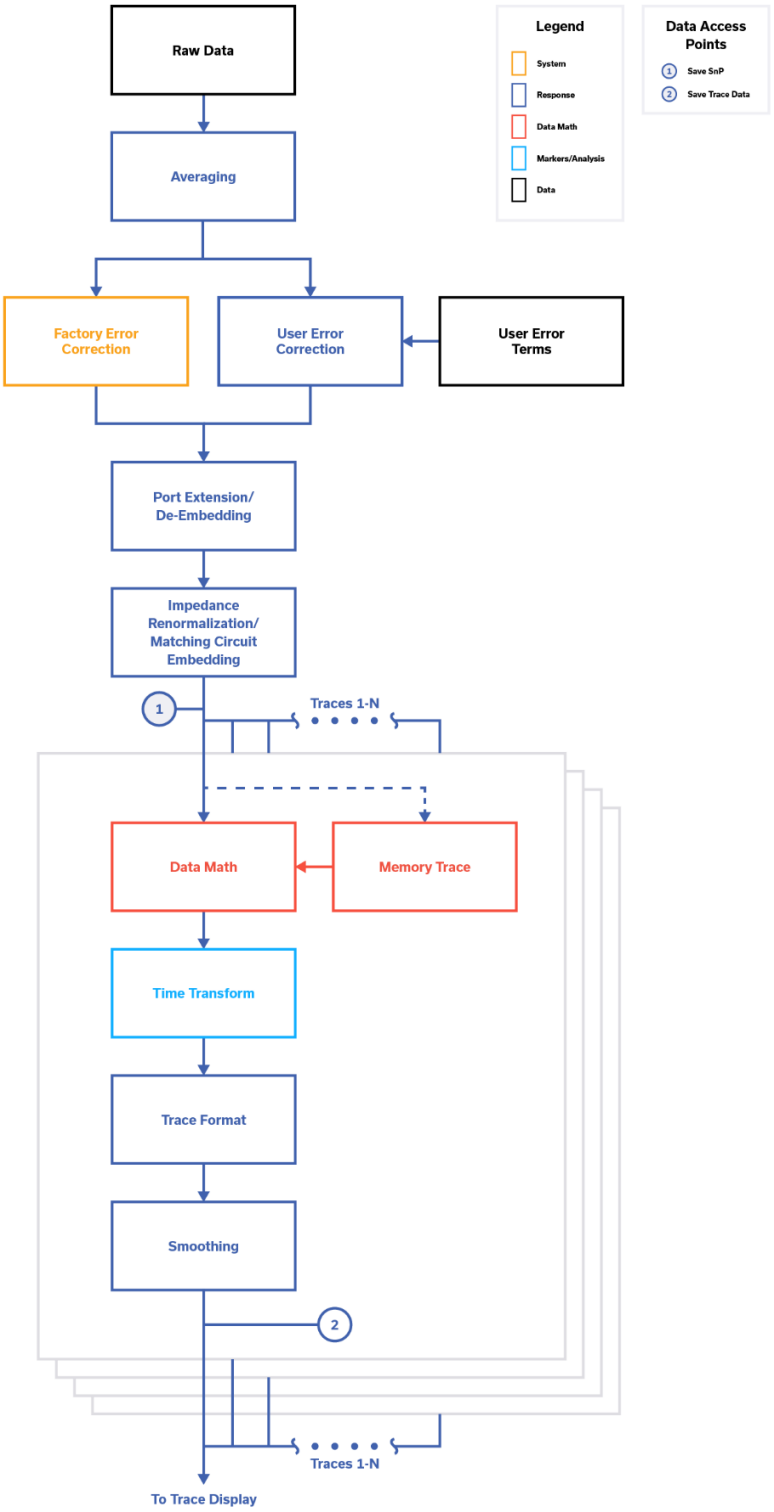
Multiple USB connected eVNA-63+ devices can be managed by a single host PC but only 1 can be operational at any time. The eVNA Studio UI and API can be used to search for available devices on the USB bus and initiate the connection with 1 instrument at a time. The first eVNA must be disconnected before a second connection with a new eVNA can be started.

For environments requiring multiple VNAs to be operational simultaneously, each eVNA should be connected by USB to its own host PC with the eVNA Studio software installed. Each eVNA system can then be controlled over a network using the API, allowing a remote workstation to control all eVNA systems simultaneously, over a network.

4. eVNA Operation

4.1. Measurement Process Flow

The figure below presents the measurement data process flow diagram for the eVNA. This summarizes the steps taken automatically by the eVNA from measurement to display and is a means to understanding where each level of data correction and manipulation is applied.



5. SCPI Overview

SCPI (Standard Commands for Programmable Instruments) is a standardized syntax for controlling test and measurement equipment. The eVNA uses SCPI commands to provide access to all VNA functions, the majority of which will be familiar to users of other SCPI controlled VNA products. Commands are sent to the eVNA using a TCP / IP connection as detailed above.

5.1. SCPI Reference Explanation

The SCPI reference sections of this manual present all available commands / queries in the following format:

Command	`CALCulate<ch>:PARameter:DEFine`
Parameters	enum
ENum Values	S11|S21|S12|S22|A|B|R1|R2
Default	S11
Scope	Trace
Summary	Set the measurement parameter for the active trace on the specified channel. Use `CALC<ch>:PAR<tr>:SEL` to set the active trace.

Example	`CALCulate:PARameter:DEFine S21`
	`CALCulate:PARameter:DEFine? ==> S21`
	`CALCulate2:PARameter:DEFine S21`
	`CALCulate2:PARameter:DEFine? ==> S21`

5.1.1. COMMANDS / QUERIES

The "Command" section of the table specifies the SCPI command (an ASCII text string) to set a parameter within the eVNA or initiate an action.

Where appropriate the same text can also be used to query the corresponding parameter by appending "?" to the end of the string.

Example: `CALCulate:PARameter:DEFine S11`

Explanation: Set the measurement parameter for the specified trace

Example: `CALCulate:PARameter:DEFine?`

Explanation: Query the measurement parameter that has been set for the specified trace

5.1.2. PARAMETERS

The "Parameters" section of the table specifies the datatype used in any arguments required by the command, or returned as the result of the query. When required, arguments can be appended to the end of the command string, separated by spaces.

Example: `CALCulate:PARameter:DEFine S11`

Explanation: Set S11 as the measurement parameter for a specific trace

The data type can be bool, double, int, string or enum (enumerated value). A list of allowed values will be specified for enum types. Where applicable, a range of values and the default setting will also be indicated.

5.1.3. ABBREVIATED COMMANDS

SCPI commands are defined using complete words to allow for easy reading but they can also be abbreviated down to the first few letters for concise programming. The abbreviated versions are indicated by the capitalized sections.

Full example: `CALCulate:PARameter:DEFine S11`

Abbreviated example: `CALC:PAR:DEF S11`

5.1.4. SCOPE OF COMMANDS / QUERIES

The "Scope" section of the table indicates whether a command applies to the instrument, channel, trace or marker.

The eVNA-63+ supports up to 16 independent channel configurations, with up to 16 display traces per channel and up to 9 markers per trace. All of these layers can be referenced and configured through the SCPI command model. The Scope of the command defines how the relevant layer has to be specified, as shown below:

Scope	Requirements
Channel	Specify the channel
Trace	Specify the channel and trace
Marker	Specify the channel, trace and marker

5.1.5. SPECIFYING CHANNELS

To work with more than 1 channel, first enable them using `SERVice:CHANnel:COUNt x`, where x is the total number of channels needed, from 1-16.

When multiple channels are in use, the channel number should be included as indicated, usually as a suffix to the first branch of the SCPI command. The suffix can be omitted when only a single channel is in use, or when channel 1 is the intended target of the command. This leads to the 3 options below for `CALCulate<ch>:PARameter:DEFine`:

1. `CALCulate:PARameter:DEFine S11` - Display S11 on channel 1 (active trace)
2. `CALCulate1:PARameter:DEFine S11` - Display S11 on channel 1 (active trace)
3. `CALCulate16:PARameter:DEFine S11` - Display S11 on channel 16 (active trace)

5.1.6. SPECIFYING TRACES

Use `CALCulate<ch>:PARameter:COUNt x` to enable additional traces on a specific channel, where x is the number of traces, from 1-16. Note that the scope of this example command is "channel" so the required number of traces should be enabled separately for each channel.

1. `CALCulate:PARameter:COUNt 4` - Enable 4 traces on channel 1
2. `:CALCulate1:PARameter:COUNt 4` - Enable 4 traces on channel 1
3. `:CALCulate16:PARameter:COUNt 8` - Enable 8 traces on channel 16

To configure an individual trace for a specific channel, the required trace number must first be selected as the active trace. The trace number is usually specified as a suffix to the second branch of the SCPI command, as indicated. Subsequent commands with a scope of "trace" will then apply to the active trace, without having to specify the trace number each time. The channel number should always be supplied (except that it can be omitted for channel 1).

The sequence below demonstrates how to select and then configure trace 4 on channel 2:

1. `CALCulate2:PARameter4:SELect` - Select trace 4 as the active trace on channel 2
2. `CALCulate2:PARameter:DEFine S21` - Display S21 on the active trace on channel 2
3. `CALCulate2:PARameter:SPORt 1` - Set source port 1 on the active trace on channel 2
4. `CALCulate2:SELected:PHASe RADians` - Set the phase units as radians for the above trace

5.1.7. SPECIFYING MARKERS

Unlike channels and traces, the number of markers does not need to be declared in advance. Any marker number, from 1 to 9, can be configured at any time by including the marker number as a suffix to "MARKer" within the SCPI command. The reference marker is a special case but can be configured in the same way by referring to marker 10.

Markers are applied to the active trace on a specified channel, so the channel number must be specified in each command and the required trace must first be selected as active on the required channel.

The sequence below demonstrates how to configure 2 markers on the active trace on channel 2:

1. `CALCulate2:SELected:MARKer1:STATe 1` - Display marker 1
2. `CALCulate2:SELected:MARKer2:STATe 1` - Display marker 2
3. `CALCulate2:SELected:MARKer1:DISCrete 1` - Fix marker 1 to discrete sweep points
4. `CALCulate2:SELected:MARKer2:DISCrete 1` - Fix marker 2 to discrete sweep points
5. `CALCulate2:SELected:MARKer1:X 2e9` - Set marker 1 at 2 GHz
6. `CALCulate2:SELected:MARKer2:X 3e9` - Set marker 2 at 3 GHz
7. `CALCulate2:SELected:MARKer1:Y?` - Read the measurement value at marker 1
8. `CALCulate2:SELected:MARKer2:Y?` - Read the measurement value at marker 2

5.2. Example SCPI Sequences

The eVNA user guide includes full descriptions of the eVNA features, along with practical instructions and important considerations to implement them. Example sequences of the SCPI commands required to implement a particular feature are also presented at the end of each user guide section. A full Python Programming Example is presented below.

6. SCPI Command Reference

This section summarizes the complete list of SCPI commands / queries supported by the eVNA.

6.1. IEEE Common Commands

Common system identification and configuration commands supported by all IEEE compliant SCPI instruments

Command	*CLS
Scope	Instrument
Summary	Clear status and event registers

Command	*ESE
Parameters	int
Range	0 to 255
Default	0
Scope	Instrument
Summary	Set the Event Status Enable Register

Command	*ESR?
Parameters	int
Scope	Instrument
Summary	Read and clear the Standard Event Status Register. See *OPC for use of *ESR? In synchronization of operations.
Example	*ESR? ==> 129

Command	*IDN?
Parameters	string, string, string, string
Scope	Instrument
Summary	Identify the eVNA as with a comma-separated list of parameters: Manufacturer, model number, serial number, firmware version number
Example	*IDN? ==> EVNA63,Mini-Circuits,21440001,Application version 3.0.8.9834e74f Firmware version A0

Command	*OPC
Scope	Instrument
Summary	Sets the OPC bit (bit 0) of the Standard Event Status Register at the completion of all pending operations. Use in conjunction with *ESR? to pause execution of any operations following *ESR? until the prior operations have completed. Note: Ensure the API socket timeout is set longer than the eVNA operation time, to avoid timeout errors.
Example	Initiate a trigger and hold all subsequent operations (*IDN?) until the sweep completes: *OPC TRIGger:SEQuence:SOURce BUS TRIGger:SEQuence:SINGle *ESR? ==> 129 *IDN? ==> EVNA63,Mini-Circuits…

Command	*OPC?
Scope	Instrument
Summary	Read the OPC bit (bit 0) of the Standard Event Status Register. The query will return after completion of all pending operations, to allow synchronisation. Note: Ensure the API socket timeout is set longer than the eVNA operation time, to avoid timeout errors.
Example	Initiate a trigger and hold all subsequent operations (*IDN?) until the sweep completes: TRIGger:SEQuence:SOURce BUS TRIGger:SEQuence:SINGle *OPC? ==> 1 *IDN? ==> EVNA63,Mini-Circuits…

Command	*OPT?
Parameters	int,…
Scope	Instrument
Summary	Output a list of identification numbers of installed options. Returns 0 if no options are installed.
Example	*OPT? ==> 0

Command	*RST
Scope	Instrument
Summary	Reset the eVNA to the factory configuration (same as:SYST:PRES)
Example	*RST

Command	*SRE
Parameters	int
Range	0 to 255
Default	0
Scope	Instrument
Summary	Set the Service Request Enable Register

Command	*STB?
Parameters	int
Scope	Instrument
Summary	Read the Status Byte Register

Command	*TRG
Scope	Instrument
Summary	Issue a trigger event if the trigger mode is set to "SCPI"
Example	*TRG

Command	*WAI
Scope	Instrument
Summary	Wait for execution of all commands before initiating any other actions
Example	Initiate a trigger and hold all subsequent operations (*IDN?) until the sweep completes: TRIGger:SEQuence:SOURce BUS TRIGger:SEQuence:SINGle *WAI *IDN? ==> EVNA63,Mini-Circuits…

6.2. Hardware Setup

6.2.1. BIAS-TEE

The bias 1 and bias 2 ports accept a DC voltage from a suitable source to be fed to the measurement ports (Port 1 and Port 2) for testing of amplifiers and other components which draw the DC supply from the RF path.

Command	SYSTem:BIAStee
Parameters	bool
Default	1
Summary	Set the bias-tee state to enable / disable the DC bias on ports 1 and 2

6.2.2. REFERENCE OSCILLATOR

The Ref In & Ref Out ports allow synchronization of measurements with external test instruments. The reference source can be switched between the internal and external (when connected) sources.

Command	SENSe:ROSCillator:AUTO
Parameters	bool
Default	0
Summary	Enable / disable automatic selection of the reference source 0 = disabled (reference source can be set by the user) 1 = enabled (external refernce will be used when a signal is detected at Ref In)

Command	SENSe:ROSCillator:LOCKed
Parameters	bool
Default	0
Summary	Check whether the reference is locked

Command	SENSe:ROSCillator:SOURce
Parameters	enum
ENum Values	INTernal|EXTernal
Default	INTernal
Scope	Instrument
Summary	Specify which reference source is used

6.3. Hardware Connection

Identify, connect and disconnect eVNA units

Command	SYSTem:CONNect
Parameters	string
Summary	Connect to an available eVNA by name

Command	SYSTem:DISConnect
Summary	Disconnect the eVNA

Command	SYSTem:DISCOVER
Summary	Provide a list of available eVNA units on the USB bus
Command	SYSTem:ISPControl:PORT
Parameters	int
Range	1 to nPorts

Default	1
Scope	Instrument
Summary	Set the initial source port

Command	SYSTem:ISPControl[:STATe]
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable initial source port control

Command	SYSTem:PRESet:MODE
Parameters	enum
ENum Values	FACtory|USER
Default	FACtory
Scope	Instrument
Summary	Set the preset mode

Command	SYSTem:SERVice?
Parameters	bool
Scope	Instrument
Summary	Read the service mode (1 = service mode, 0 = normal operation)

Command	SYSTem:TEMPerature:DATA?
Summary	Read the internal temperature sensor data (returns a reading in degrees Celcius)

Command	SYSTem:TEMPerature[:STATe]?
Parameters	bool
Summary	Check whether the eVNA is warmed up (0 = not warmed up, 1 = warmed up)

Command	SYSTem:UPReset:FILEname
Parameters	string
Default	""
Scope	Instrument
Summary	Set the user preset mode from a file

6.4. Channels & Traces

The eVNA supports up to 16 measurement channels with up to 16 traces per channel. Measurement and formatting operations are applied to specific channels and traces, so it is important to make sure the relevant channel / trace is indicated where required.

For the following examples:

- `<ch>` is the channel number from 1 to 16 (the default is channel 1 if omitted)
- `<tr>` is the trace number from 1 to 16 (the default is trace 1 if omitted)

The channel number should be specified in each command when multiple channels are in use, unless the operation is intended for channel 1. To operate on a specific trace, the trace must first be selected using `SERVice:CHANnel<ch>:TRACe:ACTive`.

Command	`CALCulate<ch>:PARameter:COUNt`
Parameters	int
Range	1 to nTraces
Default	1
Scope	Channel
Summary	Set the number of traces for the specified channel

Example	`CALCulate:PARameter:COUNt 4`
	`CALCulate:PARameter:COUNt? ==> 4`
	`CALCulate2:PARameter:COUNt 4`
	`CALCulate2:PARameter:COUNt? ==> 4`

Command	`CALCulate<ch>:PARameter<tr>:SELect`
Scope	Channel
Summary	Set the active trace for the specified channel
Example	`CALCulate:PARameter4:SELect` `CALCulate2:PARameter4:SELect`

Command	`CALCulate<ch>:PARameter:DEFine`
Parameters	enum
ENum Values	S11|S21|S12|S22|A|B|R1|R2
Default	S11
Scope	Trace
Summary	Set the measurement parameter for the active trace on the specified channel. Use `CALC<ch>:PAR<tr>:SEL` to set the active trace.

Example	`CALCulate:PARameter:DEFine S21`
	`CALCulate:PARameter:DEFine? ==> S21`
	`CALCulate2:PARameter:DEFine S21`
	`CALCulate2:PARameter:DEFine? ==> S21`

Command	CALCulate<ch>:PARameter:RPORt
Parameters	Int
Range	1 to nPorts
Default	1
Scope	Trace
Summary	Set the receiver port for the active trace on the specified channel. Use CALC<ch>:PAR<tr>:SEL to specify the active trace.
Example	CALCulate:PARameter:RPORt 2 CALCulate:PARameter:RPORt? ==> 2 CALCulate2:PARameter:RPORt 2 CALCulate2:PARameter:RPORt? ==> 2

Command	CALCulate<ch>:PARameter:SPORt
Parameters	int
Range	1 to nPorts
Default	1
Scope	Trace
Summary	Set the source port for the active trace on the specified channel. Use CALC<ch>:PAR<tr>:SEL to specify the active trace.
Example	CALCulate:PARameter:SPORt 1 CALCulate:PARameter:SPORt? ==> 1 CALCulate2:PARameter:SPORt 1 CALCulate2:PARameter:SPORt? ==> 1

Command	CALCulate<ch>[:SELected]:FORMat
Parameters	enum
ENum Values	MLOGarithmic|PHASe|GDELay|SLINear|SLOGarithmic|SCOMplex|SMITh|SADMittance|PLINear|PLOGarithmic|POLar|MLINear|SWR|REAL|IMAGinary|UPHase|PPHase
Default	MLOGarithmic
Scope	Trace
Summary	Set the data format for the active trace on the specified channel. Use CALC<ch>:PAR<tr>:SEL to specify the active trace.
Example	CALCulate:SELected:FORMat SMITh CALCulate:SELected:FORMat? ==> SMITh CALCulate2:SELected:FORMat SMITh CALCulate2:SELected:FORMat? ==> SMITh

Command	CALCulate<ch>[:SELected]:PHASe
Parameters	enum
ENum Values	DEGrees|RADians
Default	DEGrees
Summary	Set the phase units for the active trace on the specified channel. Use CALC<ch>:PAR<tr>:SEL to specify the active trace.
Example	CALCulate:SELected:PHASe RADians CALCulate:SELected:PHASe? ==> RADians CALCulate2:SELected:PHASe RADians CALCulate2:SELected:PHASe? ==> RADians

Command	DISPlay:CHANnel<ch>:ACTivate
Scope	Instrument
Summary	Specify which of the allocated channels is the active channel. Note: The specified channel number must be within the range of allocated channels.
Example	DISPlay:CHANnel2:ACTivate DISPlay:CHANnel:ACTivate? ==> 2

Command	DISPlay:CHANnel<ch>:SPLit
Parameters	enum
ENum Values	D1|D12|D1_2|D112|D1_1_2|D123|D1_2_3|D12_33|D11_23|D13_23|D12_13|D1234| D1_2_3_4|D12_34|D123_456|D12_34_56|D1234_5678|D12_34_56_78|D123_456_789| D1234__9ABC|D123__ABC|D1234__DEFG
Default	D1
Scope	Channel
Summary	Set the trace layout for the specified channel, from the available presets. Underscores in the arguments indicate a new row. Examples: D1 = Single graph D12 = 2 graphs, side by side D1_2 = 2 graphs, one on top of the other D123_456_789 = 3 rows, each of 3 graphs
Example	DISPlay:CHANnel:SPLit D12_34 DISPlay:CHANnel:SPLit? ==> D12_34 DISPlay:CHANnel2:SPLit D12_34 DISPlay:CHANnel2:SPLit? ==> D12_34

Command	DISPlay:CHANnel<ch>:TRACe<tr>[:STATe]
Parameters	bool
Default	1
Scope	Trace
Summary	Enable / disable the display of the specified trace on the specified channel
Example	DISPlay:CHANnel:TRACe:STATe 0 DISPlay:CHANnel:TRACe:STATe? ==> 0 DISPlay:CHANnel2:TRACe3:STATe 0 DISPlay:CHANnel2:TRACe3:STATe? ==> 0

Command	DISPlay:SPLit
Parameters	enum
ENum Values	D1|D12|D1_2|D112|D1_1_2|D123|D1_2_3|D12_33|D11_23|D13_23|D12_13|D1234| D1_2_3_4|D12_34|D123_456|D12_34_56|D1234_5678|D12_34_56_78|D123_456_789| D1234__9ABC|D123__ABC|D1234__DEFG
Default	D1
Scope	Instrument
Summary	Set the channel layout from the available presets. Underscores in the arguments indicate a new row. Examples: D1 = Single graph D12 = 2 graphs, side by side D1_2 = 2 graphs, one on top of the other D123_456_789 = 3 rows, each of 3 graphs
Example	DISPlay:SPLit D12_34 DISPlay:SPLit? ==> D12_34

Command	SERVice:CHANnel:ACTive
Parameters	int
Range	1 to number of allocated channels
Default	1
Scope	Instrument
Summary	Specify which of the allocated channels is the active channel. Note: The specified channel number must be within the range of allocated channels.
Example	SERVice:CHANnel:ACTive 2 SERVice:CHANnel:ACTive? ==> 2

Command	SERVice:CHANnel:COUNt
Parameters	int
Range	1 to max Channels
Default	1
Scope	Instrument
Summary	Set the number of enabled channels
Example	SERVice:CHANnel:COUNt 4 SERVice:CHANnel:COUNt? ==> 4

Command	SERVice:CHANnel<ch>:TRACe:ACTive
Parameters	int
Range	1 to max traces
Scope	Channel
Summary	Set the active trace for the specified channel
Example	SERVice:CHANnel:TRACe:ACTive 3 SERVice:CHANnel:TRACe:ACTive? ==> 3 SERVice:CHANnel2:TRACe:ACTive 3 SERVice:CHANnel2:TRACe:ACTive? ==> 3

Command	SERVice:CHANnel<ch>:TRACe:COUNt?
Parameters	int
Range	1 to max traces
Scope	Channel
Summary	Check the number of enabled traces for the specified channel
Example	SERVice:CHANnel:TRACe:COUNt? ==> 3 SERVice:CHANnel2:TRACe:COUNt? ==> 3

6.5. Frequency & Power Stimulus

The stimulus settings define the frequency and power of the output ports and any parameters related to the measurement sweep.

Command	OUTPut[:STATe]
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable the stimulus outputs required for measurements. Must be enabled after a power failure.

Command	SENSe
Parameters	[int]
Range	1 to nChannels
Default	1
Summary	Select the channel number

Command	SENSe:SWEep:POINts
Parameters	int
Scope	Channel
Summary	Set the number of measurement points for the active channel

Command	SENSe:SWEep:TYPE
Parameters	enum
ENum Values	LINear|LOGarithmic|SEGMent|POWer
Default	LINear
Scope	Channel
Summary	Set the sweep type for the active channel

6.5.1. FREQUENCY SWEEP

The eVNA can operate a frequency sweep with linear or logarithmic values plotted. The default condition is a linear frequency sweep

Command	SENSe:FREQuency:CENTer
Parameters	double
Range	fMin to fMax
Default	fMin
Units	Hz
Scope	Channel
Summary	Set the sweep center frequency

Command	SENSe:FREQuency:DATA?
Parameters	double,...
Units	Hz
Scope	Channel
Summary	Read an array frequency values, one for each measurement point

Command	SENSe:FREQuency:SPAN
Parameters	double
Range	0 to fMax
Default	fMin
Units	Hz
Scope	Channel
Summary	Sweep frequency span

Command	SENSe:FREQuency:STARt
Parameters	double
Range	fMin to fMax
Default	fMin
Units	Hz
Scope	Channel
Summary	Set the sweep start frequency

Command	SENSe:FREQuency:STOP
Parameters	double
Range	fMin to fMax
Default	fMin
Units	Hz
Scope	Channel
Summary	Set the sweep stop frequency

6.5.2. POWER SWEEP

Setting the stimulus to power sweep mode allows measurement of a DUT response to increasing power level, for example to plot the compression point of an amplifier.

Command	SENSe:FREQuency[:CW|FIXed]
Parameters	double
Range	fMin to fMax
Default	fMin
Units	Hz
Scope	Channel
Summary	Set the CW frequency value for the power sweep

Command	SOURce
Parameters	[int]
Range	1 to nChannels
Default	1
Summary	Channel number

Command	SOURce:POWer:CENTer
Parameters	double
Range	pMin to pMax
Default	-15
Units	dBm
Scope	Channel
Summary	Set the center value for a power sweep

Command	SOURce:POWer:PORT
Parameters	[int]
Range	1 to nPorts
Default	1
Summary	Port number

Command	SOURce:POWer:PORT:COUPle
Parameters	bool
Default	1
Scope	Channel
Summary	Turn on to set the same power level at all ports, turn off to allow different power levels on each port

Command	SOURce:POWer:PORT[:LEVel][:IMMediate][:AMPLitude]
Parameters	double
Range	pMin to pMax
Default	-2
Units	dBm
Scope	Channel, port
Summary	Set the power level per port
Command	SOURce:POWer:SPAN
Parameters	double
Range	pMin to pMax

Default	15
Units	dBm
Scope	Channel
Summary	Set the span value for the power sweep

Command	SOURce:POWer:STARt
Parameters	double
Range	pMin to pMax
Default	-32
Units	dBm
Scope	Channel
Summary	Set the start value for power sweep

Command	SOURce:POWer:STOP
Parameters	double
Range	pMin to pMax
Default	-2
Units	dBm
Scope	Channel
Summary	Set the stop value for power sweep

Command	SOURce:POWer[:LEVel]:SLOPe:STATe
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable the power slope correction

Command	SOURce:POWer[:LEVel]:SLOPe[:DATA]
Parameters	double
Range	-2 to 2
Default	0
Units	dB/GHz
Scope	Channel
Summary	Correct the power slope

Command	SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]
Parameters	double
Scope	Channel
Summary	Set the current power level

6.5.3. SEGMENTED FREQUENCY SWEEP

Segmented sweeps allow a compromise to be made between data resolution and overall sweep time, by setting the measurement points in a sweep to be focused at the particular frequency ranges of the DUT characteristic where the highest resolution is required.

Command	MMEMory:LOAD:SEGMent
Parameters	string
Scope	Channel

Summary	Load the segment table from .csv file

Command	MMEMory:STORe:SEGMent
Parameters	string
Scope	Channel
Summary	Store the segment table in .csv file

Command	SENSe:SEGMent:DATA
Parameters	double,…
Scope	Channel
Summary	Set the segment sweep table with an array of values: <span_mode>,<ifbw_mode>,<power_mode>,<delay_mode>, <sweep_mode>,<time_mode>,<segments>, <start n>,<stop n>,<points n>,<ifbw n>,<power n>,<delay n>,<time n>,…, span_mode = 1 (start/stop) or 0 (center/span) ifbw_mode = set IF bandwidth per segment (bool 0 or 1) power_mode = set power per segment (bool 0 or 1) delay_mode = set sweep delay per segment (bool 0 or 1) sweep_mode = set sweep mode per segment (bool 0 or 1) time_mode = set sweep time per segment (bool 0 or 1) segments = number of segments to specify start n = start / center value for segment n stop n = stop / span value for segment n points n = number of points for segment n ifbw n = ifbw for segment n (if required) power n = power for segment n (if required) delay n = sweep delay for segment n (if required) sweep n = sweep mode per segment n (if required), Stepped = normal sweep or FStepped = fast sweep mode time n = sweep time for segment n (if required)

Command	SENSe:SEGMent:SWEep:POINts?
Parameters	int
Scope	Channel
Summary	Read total number of points for segmented sweep

Command	SENSe:SEGMent:SWEep:TIME[:DATA]?
Parameters	double
Scope	Channel
Summary	Read total sweep time for segmented sweep

6.5.4. ADVANCED SWEEP SETTINGS

Command	SENSe:BANDwidth
Parameters	int
Range	1 to 500000
Default	10000
Units	Hz
Scope	Channel
Summary	Set the IF bandwidth for the active channel

Command	SENSe:SWEep:DELay
Parameters	double
Range	0 to 1
Default	0
Units	sec
Scope	Channel
Summary	Set the sweep delay for the active channel

Command	SENSe:SWEep:GENeration
Parameters	enum
ENum Values	STEPped|FSTepped
Default	STEPped
Scope	Channel
Summary	Set the sweep mode for the active channel: Stepped = normal sweep mode FStepped = fast sweep mode

Command	SENSe:SWEep:TIME:AUTO
Parameters	bool
Default	1
Scope	Channel
Summary	Enable / disable auto sweep time

Command	SENSe:SWEep:TIME[:DATA]
Parameters	double
Units	sec
Scope	Channel
Summary	Set the sweep time for the active channel (automatically corrected if less than the minimum time)

6.5.5. SWEEP TRIGGERS

Measurement sweeps commence in response to a trigger signal. By default, the eVNA is set to the internal trigger source in continuous mode, meaning sweeps repeat continuously.

Command	ABORt

Scope	Instrument
Summary	Abort the current measurements (for all channels)
Example	`ABORt`

Command	`INITiate:CONTinuous`
Parameters	bool
Range	0 to 1
Scope	Channel
Summary	Sets the trigger mode between Internal / Continuous (1) or Manual / Hold (0)
Example	`INITiate:CONTinuous 1` `INITiate:CONTinuous? ==> 1`

Command	`INITiate[:IMMediate]`
Scope	Channel
Summary	If the trigger for the channel is idle and continuous=false, send a trigger signal and start a measurement. Otherwise, error.
Example	`INITiate:IMMEDiate`

Command	`TRIGger:[SEQuence]:POINt`
Parameters	bool
Default	0
Scope	Instrument
Summary	Toggle the point trigger mode on (single point per trigger) or off (full sweep per trigger)
Example	`TRIGger:SEQuence:POINt 1` `TRIGger:SEQuence:POINt? ==> 1`

Command	`TRIGger:[SEQuence]:SCOPe`
Parameters	enum
ENum Values	ALL|ACTive
Default	ALL
Scope	Instrument
Summary	Set the trigger scope between all channels or the active channel
Example	`TRIGger:SEQuence:SCOPe ACTive` `TRIGger:SEQuence:SCOPe? ==> ACTive`

Command	TRIGger:[SEQuence]:SINGle
Scope	Instrument
Summary	Generates a trigger signal to start the measurement immediately. The trigger source must first be set to TRIG:SOUR BUS. The command completes after the measurement has finished (use *OPC? to query for the end of the measurement sequence).
Example	TRIGger:SEQuence:SOURce BUS INITiate:CONTinuous 1 TRIGger:SEQuence:SINGle

Command	TRIGger:[SEQuence]:SOURce
Parameters	enum
ENum Values	INTernal|EXTernal|MANual|BUS
Default	INTernal
Scope	Instrument
Summary	Set the trigger source: 1. Internal – Default mode, sweeps run continuously 2. External – Sweeps are triggered by a TTL signal on the Trig In port 3. Manual – Sweeps are triggered using the Trigger button in the eVNA Studio GUI 4. Bus – Sweeps are triggered using the SCPI command TRIG:SING. Internal / continuous mode should also be set using INIT:CONT 1.
Example	INITiate:CONTinuous 1 TRIGger:SEQuence:SOURce BUS TRIGger:SEQuence:SOURce? ==> BUS

Command	TRIGger:[SEQuence][:IMMediate]
Scope	Instrument
Summary	Generate trigger and start measurement immediately when the trigger source is in manual or SCPI mode. The command completes immediately. Internal / continuous mode should also be set using INIT:CONT 1.
Example	INITiate:CONTinuous 1 TRIGger:SEQuence:IMMediate

Command	TRIGger:AVERage
Parameters	bool
Default	0
Scope	Instrument
Summary	Enable / disable averaging trigger. When enabled, multiple sweeps are performed for each trigger to average the results.
Example	SENSe:AVERage:COUNt 8 TRIGger:AVERage 1 TRIGger:AVERage? ==> 1

Command	TRIGger:EXTernal:DELay
Parameters	double
Range	0 to 1
Default	0
Units	sec
Scope	Instrument
Summary	Set a time delay for all channels between an external trigger signal and starting measurements.
Example	TRIGger:EXTernal:DELay 5e-9 TRIGger:EXTernal:DELay? ==> 5e-09

6.6. Measurement & Data Formatting

Configuration of trace displays, scaling, data types and measurement averaging.

6.6.1. ELECTRICAL DELAY

Configure a custom electrical delay for the measured data.

Command	CALCulate[:SELected]:CORRection:EDELay:MEDium
Parameters	enum
ENum Values	COAXial|WAVeguide
Default	COAX
Scope	Channel
Summary	Set the media type for electrical delay time calculation

Command	CALCulate[:SELected]:CORRection:EDELay:TIME
Parameters	double
Range	-10 to 10
Default	sec
Units	0
Scope	Trace
Summary	Set the electrical delay time for the current trace

Command	CALCulate[:SELected]:CORRection:EDELay:WGCutoff
Parameters	double
Range	fMin to fMax
Default	fMin
Units	Hz
Scope	Channel
Summary	Set the waveguide cutoff frequency for the current trace when wavegude is the media type

Command	CALCulate[:SELected]:CORRection:OFFSet:PHASe
Parameters	double
Range	-360 to 360
Default	0
Units	degrees
Scope	Trace
Summary	Set a phase offset on the current trace

Command	SENSe:CORRection:RVELocity:COAX
Parameters	double
Range	0 to 10
Default	1
Scope	Channel
Summary	Set or query the velocity factor for the active channel (1/seqr(epsr))

6.7. Averaging

6.7.1. SWEEP AVERAGING

With sweep averaging enabled, each discrete data point in a sweep is calculated as an average across a set number of sweeps, therefore minimizing the effect of any random noise fluctuations on the data.

Command	SENSe:AVERage:CLEar
Scope	Channel
Summary	Clear the previous averaging data and restart averaging

Command	SENSe:AVERage:COUNt
Parameters	int
Range	1 to 999
Default	16
Scope	Channel
Summary	Set the averaging factor
Example	SENSe:AVERage:COUNt 8 SENSe:AVERage:COUNt? ==> 8

Command	SENSe:AVERage[:STATe]
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable averaging

6.7.2. TRACE SMOOTHING

Trace smoothing is distinct from sweep averaging. A moving average is calculated from consecutive points along a single sweep in order to smooth out sudden variations in signal levels.

Command	CALCulate[:SELected]:SMOothing:APERture
Parameters	double
Range	0.05 to 25
Default	1.5
Units	%
Scope	Trace
Summary	Set the moothing apperture for the active trace as a percentage of the sweep span

Command	CALCulate[:SELected]:SMOothing[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable trace smoothing

6.8. Markers

Markers provide a flexible set of options for highlighting significant data points and statistics, performing calculations and searching for particular signal characteristics. The eVNA supports up to 9 markers per trace, plus a reference marker.

<mk> is the marker number in the following examples (1 to 9 for the standard markers, or 10 for the reference marker).

Command	CALCulate[:SELected]:MARKer<mk>:ACTivate
Summary	Set the specified marker as the active marker
Example	CALCulate:SELected:MARKer1:ACTivate

Command	CALCulate[:SELected]:MARKer<mk>:COUPle
Parameters	bool
Default	1
Scope	Channel
Summary	Set whether the specified marker should be coupled to all traces: 0 = Marker is independent 1 = Marker is coupled across all traces on the specified channel
Example	CALCulate:SELected:MARKer1:COUPle 0 CALCulate:SELected:MARKer1:COUPle? ➔ 0

Command	CALCulate[:SELected]:MARKer<mk>:DISCrete
Parameters	bool
Default	1
Scope	Trace
Summary	Enable any sweep value for the specified marker, or fix to the discrete measurement points: 0 = Discrete mode off (marker can move between measurement sweep points) 1 = Discrete mode on (marker fixes to nearest sweep point)
Example	CALCulate:SELected:MARKer1:DISCrete 0 CALCulate:SELected:MARKer1:DISCrete? ➔ 1

Command	CALCulate[:SELected]:MARKer:REFerence[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable reference mode for a specific marker. Reference mode converts the marker labels to show the delta values relative to the reference marker. Use marker 10 to set the position of the reference marker, then enable reference mode on each of markers 1-9 as required.
Example	CALCulate:SELected:MARKer10:STATe 1 CALCulate:SELected:MARKer10:X 3e9 CALCulate:SELected:MARKer1:STATe 1 CALCulate:SELected:MARKer1:X 2e9 CALCulate:SELected:MARKer1:REFerence:STATe 1 CALCulate:SELected:MARKer1:REFerence:STATe? ➔ 1

Command	CALCulate[:SELected]:MARKer<mk>:SET
Parameters	enum
ENum Values	STARt|STOP|CENTer|RLEVel|DELay
Scope	Trace
Summary	Set the specified stimulus value based on the active marker position (start, stop, center, reference level, electrical delay time)
Example	CALCulate:SELected:MARKer1:SET CENTer

Command	CALCulate[:SELected]:MARKer<mk>:X
Parameters	double
Default	Sweep start frequency
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Set the stimulus value for the specified marker using scientific notation
Example	CALCulate:SELected:MARKer1:X 1e9 CALCulate:SELected:MARKer1:X? ➔ 1e+09

Command	CALCulate[:SELected]:MARKer<mk>:Y?
Parameters	double, double
Scope	Trace
Summary	Read the response value for the specified marker. Two data values are returned (primary, secondary) to allow for Smith / polar formats, the second data value will be 0 for other formats.
Example	CALCulate:SELected:MARKer1:Y? ➔ (-9.29212,0)

Command	CALCulate[:SELected]:MARKer<mk>[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable display of the specified marker
Example	CALCulate:SELected:MARKer1:STATe 1 CALCulate:SELected:MARKer1:STATe? ➔ 1

Command	DISPlay:CHANnel:ANNotation:MARKer:ALIGn[:STATe]
Parameters	bool
Default	1
Scope	Channel
Summary	Configure location of marker information with trace 1: 1 (On) = Display info above the active marker 0 (Off) = Display info in top left corner

Command	DISPlay:CHANnel:ANNotation:MARKer:SINGle[:STATe]
Parameters	bool
Default	1
Scope	Trace
Summary	Configure which marker information is displayed 1 (On) = Display marker values for active trace only 0 (Off) = Display marker values for all traces

Command	DISPlay:CHANnel:TRACe:ANNotation:MARKer:POSition:X
Parameters	int
Range	-15 to 100
Default	1
Units	%
Scope	Trace
Summary	X position of marker value (as a percentage of display width)

Command	DISPlay:CHANnel:TRACe:ANNotation:MARKer:POSition:Y
Parameters	int
Range	-15 to 100
Default	1
Units	%
Scope	Trace
Summary	Y position of marker value (as a percentage of display height)

6.9. Marker Searches

Allows specific data points and frequencies to be highlighted automatically based on characteristics of the measured data.

Command	CALCulate[:SELected]:MARKer:FUNCtion:DOMain:COUPle
Parameters	bool
Default	1
Scope	Channel
Summary	Specify whether the marker search range should be coupled to all traces: 0 = Marker search range applies to the active trace only 1 = Marker search range is coupled to all traces on the specified channel

Command	CALCulate[:SELected]:MARKer:FUNCtion:DOMain:STARt
Parameters	double
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Start value of the marker search range

Command	CALCulate[:SELected]:MARKer:FUNCtion:DOMain:STOP
Parameters	double
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Stop value of the marker search range

Command	CALCulate[:SELected]:MARKer:FUNCtion:EXECute
Scope	Trace
Summary	Execute the marker search

Command	CALCulate[:SELected]:MARKer:FUNCtion:TRACking
Parameters	bool
Default	0
Summary	Enable / disable marker tracking

Command	CALCulate[:SELected]:MARKer:FUNCtion:TYPE
Parameters	enum
ENum Values	MAXimum|MINimum|PEAK|LPEak|RPEak|TARGet|LTARget|RTARget
Default	MAXimum
Summary	Set the marker search type for the active trace (max value, min value, peak, peaks to the left, peaks to the right, target, targets to the left, targets to the right)

6.9.1. PEAK SEARCH

Search the next peak within the data (based on the current marker position)

Command	CALCulate[:SELected]:MARKer:FUNCtion:PEXCursion
Parameters	double
Range	0 to 500000000
Default	3

Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the lower limit for the peak excursion marker search

Command	CALCulate[:SELected]:MARKer:FUNCtion:PPOLarity
Parameters	enum
ENum Values	POSitive|NEGative|BOTH
Default	POSitive
Summary	Set the polarity for the peak excursion marker search

6.9.2. TARGET SEARCH

Search the next target value within the data (based on the current marker position)

Command	CALCulate[:SELected]:MARKer:FUNCtion:TARGet
Parameters	double
Range	-500000000 to 500000000
Default	0
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the target value for the marker target search (moves the marker to the closest value on the active trace)

Command	CALCulate[:SELected]:MARKer:FUNCtion:TTRansition
Parameters	enum
ENum Values	POSitive|NEGative|BOTH
Default	BOTH
Summary	Set the transition type for a target search

6.9.3. BANDWIDTH SEARCH

Identifies the pass-band of the trace (based on the current marker position) to display cut-off frequencies, bandwidth and Q value.

Command	CALCulate[:SELected]:MARKer:BWIDth:DATA?
Parameters	double, double, double, double
Scope	Trace
Summary	Return the bandwidth search results for the active marker (bandwidth, center frequency, Q value, loss)

Command	CALCulate[:SELected]:MARKer:BWIDth:THReshhold
Parameters	double
Range	-500000000 to 500000000
Default	3
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the bandwidth value

Command	CALCulate[:SELected]:MARKer:BWIDth[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable display of the bandwidth search result

6.9.4. NOTCH SEARCH

Searches for a notch to the left of the current marker to display cut-off frequencies, bandwidth and Q value.

Command	CALCulate[:SELected]:MARKer:NOTCh:DATA?
Parameters	double, double, double, double
Scope	Trace
Summary	Return the results of a notch search (bandwidth, center frequency, Q value, loss)

Command	CALCulate[:SELected]:MARKer:NOTCh:THReshhold
Parameters	double
Range	-500000000 to 500000000
Default	-3
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the notch value definition

Command	CALCulate[:SELected]:MARKer:NOTCh[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable notch search

Command	CALCulate[:SELected]:MARKer:FUNCtion:DOMain[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Marker search range: 1 = arbitrary range, 0 = entire sweep range Specify whether the marker search range should be applied to the abitrary range (set by the START and STOP parameters) or the entire sweep range: 0 = entire sweep range 1 = arbitrary range

6.10. Data Searches

Allows specific data points and frequencies to be highlighted automatically based on characteristics of the measured data (similar to marker searches but returning the result rather than displaying on the marker annotation)

Command	CALCulate[:SELected]:FUNCtion:DATA?
Parameters	double,...
Summary	Returns the data array for the configured statsitical analysis, with a pair of data values for each measurement point (response and stimulus). The stimulus value will be 0 for peak to peak / stdev / mean analysis type. array[1] = Point 1 response value array[2] = Point 2 stimulus value ... array[2n-1] = Point n response value array[2n] = Point n stimulus value

Command	CALCulate[:SELected]:FUNCtion:DOMain:COUPle
Parameters	bool
Default	1
Scope	Channel
Summary	Specify whether the analysis range should be coupled to all traces: 0 = Range applies to the active trace only 1 = Range is coupled to all traces on the specified channel

Command	CALCulate[:SELected]:FUNCtion:DOMain:STARt
Parameters	double
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Set the start frequency for the statistical analysis

Command	CALCulate[:SELected]:FUNCtion:DOMain:STOP
Parameters	double
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Set the stop frequency for the statistical analysis

Command	CALCulate[:SELected]:FUNCtion:DOMain[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Specify whether the statistical analysis shuold be applied to the abitrary range (set by the START and STOP parameters) or the entire sweep range: 0 = entire sweep range 1 = arbitrary range
Command	CALCulate[:SELected]:FUNCtion:EXECute
Scope	Trace

Summary	Execute the statistical analysis

Command	CALCulate[:SELected]:FUNCtion:PEXCursion
Parameters	double
Range	0 to 500000000
Default	3
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the lower limit for the peak excursion search

Command	CALCulate[:SELected]:FUNCtion:POINts?
Parameters	int
Scope	Trace
Summary	Return the number of data pairs in the statistical analysis result

Command	CALCulate[:SELected]:FUNCtion:PPOLarity
Parameters	enum
ENum Values	POSitive|NEGative|BOTH
Default	POSitive
Scope	Trace
Summary	Set the polarity for the peak excursion search

Command	CALCulate[:SELected]:FUNCtion:TARGet
Parameters	double
Range	-500000000 to 500000000
Default	0
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the value for a target search on the active trace

Command	CALCulate[:SELected]:FUNCtion:TTRansition
Parameters	enum
ENum Values	POSitive|NEGative|BOTH
Default	BOTH
Scope	Trace
Summary	Set the transition type for a target search

Command	CALCulate[:SELected]:FUNCtion:TYPE
Parameters	enum
ENum Values	PTPeak|STDEV|MEAN|MAXimum|MINimum|PEAK|APEak|ATARget
Default	PTPeak
Scope	Trace
Summary	Set the statistical analysis type for the active trace (peak to peak / stdev / mean / max / min / peak / all peaks / all targets)

6.11. Measurement Pass / Fail Tests

Automated pass / fail testing of measured data

6.11.1. SPECIFICATION LIMIT TEST

The limit test allows a number of frequency bands to be defined with minimum or maximum test specifications per band, displaying a failure warning if required.

Command	CALCulate[:SELected]:LIMit:DATA
Parameters	double,...
Scope	Trace
Summary	Set the table of limit lines for the active trace using a data array, the first value indicating the number of limit lines, followed by groups of 5 values to define each line: array[0] = Number of limit lines array[5n-4] = Line n type (0 = off, 1 = upper limit, 2 = lower limit) array[5n-3] = x coordinate of line n start point array[5n-2] = x coordinate of line n end point array[5n-1] = y coordinate of line n start point array[5n] = y coordinate of line n end point

Command	CALCulate[:SELected]:LIMit:DISPlay[:STATe]
Parameters	bool
Units	0
Scope	Trace
Summary	Enable / disable display of limit lines for the active channel

Command	CALCulate[:SELected]:LIMit:FAIL?
Parameters	bool
Scope	Trace
Summary	Return the result of the limit test for the active channel: 0 = pass 1 = fail

Command	CALCulate[:SELected]:LIMit:OFFSet:AMPLitude
Parameters	double
Range	-500000000 to 500000000
Default	0
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the limit line amplitude offset value

Command	CALCulate[:SELected]:LIMit:OFFSet:MARKer
Scope	Trace
Summary	Set the active marker as the limit line amplitude offset value

Command	CALCulate[:SELected]:LIMit:OFFSet:STIMulus
Parameters	double
Range	-1000000000000 to 1000000000000
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Set the limit line stimulus offset value

Command	CALCulate[:SELected]:LIMit:REPort:ALL?
Parameters	double,…
Summary	Return a data array of limit test results for each measurement point on the active trace, with 4 values per result: array[4n-3] = stimulus value array[4n-2] = 0 (fail), 1 (pass) or -1 (no limit) array[4n-1] = upper limit value (or 0 if no limit specified) array[4n] = lower limit value (or 0 if no limit specified)

Command	CALCulate[:SELected]:LIMit:REPort:POINts?
Parameters	int
Summary	Return the number of measurement points that failed the limit test

Command	CALCulate[:SELected]:LIMit:REPort[:DATA]?
Parameters	double,…
Summary	Return a list of all measurement stimulus values n the active trace which failed the limit test

Command	CALCulate[:SELected]:LIMit[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable limit test

Command	MMEMory:LOAD:LIMit
Parameters	string
Scope	Trace
Summary	Load the limit table from .csv file

Command	MMEMory:STORe:LIMit
Parameters	string
Scope	Trace
Summary	Store the limit table in .csv file

6.11.2. RIPPLE LIMIT TEST

Allows a number of frequency bands to be defined with maximum ripple specifications per band, displaying a failure warning if required.

Command	CALCulate[:SELected]:RLIMit:DATA
Parameters	double,...
Summary	Set the table of ripple limit bands for the active trace using a data array, the first value indicating the number of limit lines, followed by groups of 4 values to define each band: array[0] = number of ripple bands array[4n-3] = ripple n type (0 = off, 1 = on) array[4n-2] = x coordinate of band n start point array[4n-1] = x coordinate of band n end point array[4n] = max ripple range (dB) of band n

Command	CALCulate[:SELected]:RLIMit:DISPlay:LINE
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable display of the ripple limit line for the active trace

Command	CALCulate[:SELected]:RLIMit:DISPlay:SELect
Parameters	int
Range	1 to nLimitLines
Default	1
Scope	Trace
Summary	Select the band to display for the ripple limit test

Command	CALCulate[:SELected]:RLIMit:DISPlay:VALue
Parameters	enum
ENum Values	OFF|ABSolute|MARgin
Default	OFF
Scope	Trace
Summary	Set the display type for the ripple value

Command	CALCulate[:SELected]:RLIMit:FAIL?
Parameters	bool
Scope	Trace
Summary	Check the result of the ripple tests on the active trace (1 = fail, 0 = pass)

Command	CALCulate[:SELected]:RLIMit:REPort[:DATA]?
Summary	Return a data array indicating the results of all ripple tests on the active trace: array[0] = number of ripple limit lines array[3n-2] = index of ripple limit band array[3n-1] = ripple value array[3n] = ripple test result (0 = pass, 1 = fail)

Command	CALCulate[:SELected]:RLIMit[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable the display of ripple limit tests

Command	MMEMory:LOAD:RLIMit
Parameters	string
Scope	Trace
Summary	Load the ripple limit table from .csv file

Command	MMEMory:STORe:RLIMit
Parameters	string
Scope	Trace
Summary	Store ripple limit table in .csv file

6.11.3. BANDWIDTH TEST

The bandwidth test identifies the pass band of the trace and compares to specified min / max limits, displaying a failure warning if required.

Command	CALCulate[:SELected]:BLIMit:DB
Parameters	double
Range	0 to 200
Default	3
Units	dB
Scope	Trace
Summary	Set the bandwidth threshold (attenuation from peak) for the bandwidth test of the specified trace

Command	CALCulate[:SELected]:BLIMit:DISPlay:MARKer
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable the marker display for the bandwidth test

Command	CALCulate[:SELected]:BLIMit:DISPlay:VALue
Parameters	bool
Default	0

Scope	Trace
Summary	Enable / disable display of the bandwidth value for the bandwidth test

Command	`CALCulate[:SELected]:BLIMit:FAIL?`
Parameters	bool
Scope	Trace
Summary	Read the bandwidth test result (1 = fail, 0 = pass)

Command	`CALCulate[:SELected]:BLIMit:MAXimum`
Parameters	double
Range	0 to fMax
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Set the upper frequency limit for the bandwidth test

Command	`CALCulate[:SELected]:BLIMit:MINimum`
Parameters	double
Range	0 to fMax
Default	0
Units	Stimulus (Hz / dBm / second)
Scope	Trace
Summary	Set the lower frequency limit for the bandwidth test

Command	`CALCulate[:SELected]:BLIMit:REPort[:DATA]?`
Parameters	double
Scope	Trace
Summary	Read the bandwidth value of the bandwidth test

Command	`CALCulate[:SELected]:BLIMit[:STATe]`
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable the bandwidth limit test

6.12. Window / Display Configuration

Configure the appearance and features displayed on the GUI.

Command	`DISPlay:ANNotation:FREQuency[:STATe]`
Parameters	bool
Default	1
Scope	Instrument
Summary	Show or hide the frequency information (for all windows)

Command	`DISPlay:ANNotation[:DATA]`
Parameters	string
Default	""
Scope	Instrument

Summary	Display text in the echo bar

Command	DISPlay:ANNotationFACTory:MODE:ENable
Scope	Instrument
Summary	Clear text in the echo bar

Command	DISPlay:ANNotationFACTory:STORe:STATe:DEFault
Parameters	string
Scope	Instrument
Summary	Serialize state into the given file name

Command	DISPlay:CHANnel:LABel
Parameters	bool
Scope	Channel
Summary	Show / hide the y axis label

Command	DISPlay:CHANnel:MAXimize
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable maximization of active trace

Command	DISPlay:CHANnel:TITLe:DATA
Parameters	string
Default	""
Scope	Channel
Summary	Set the title / label for this channel

Command	DISPlay:CHANnel:TITLe[:STATe]
Parameters	bool
Default	0
Scope	Channel
Summary	Show / hide the channel title / label

Command	DISPlay:CHANnel:TRACe:Y[:SCALe]:AUTO
Scope	Trace
Summary	Auto scale the y axis for the current trace

Command	DISPlay:CHANnel:TRACe:Y[:SCALe]:PDIVision
Parameters	double
Range	1E-18 to 100000000
Default	Log mag: 10 Phase: 90 Group delay: 1e-8 Smith, polar, SWR: 1 Lin mag: 0.1 Real, Imag: 0.2
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the scale per division for the current trace

Command	DISPlay:CHANnel:TRACe:Y[:SCALe]:RLEVEL
Parameters	double
Range	-0.00000005 to 0.00000005
Units	Data format (dB / degree / sec / number)
Scope	Trace
Summary	Set the reference level for the current trace to a specific value

Command	DISPlay:CHANnel:TRACe:Y[:SCALe]:RPOSition
Parameters	int
Summary	Set the refence level for the current trace to a specific division line

Command	DISPlay:CHANnel:X:SPACing
Parameters	enum
ENum Values	LINear|OBASe
Default	OBASe
Scope	Channel
Summary	Set the x axis type (linear frequency / point order base)

Command	DISPlay:CHANnel:Y[:SCALe]:DIVisions
Parameters	int
Range	4 to 30
Default	10
Scope	Channel
Summary	Set the number of Y axis divisions

Command	DISPlay:ENABle
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable display update (disabling frees up resources for making measurements)

Command	DISPlay:FONT:GRAPh
Parameters	int
Range	8 to 20
Default	9
Scope	Instrument
Summary	Font size of the graph text

Command	DISPlay:FONT[:BAR]
Parameters	int
Range	8 to 20
Default	11
Scope	Instrument
Summary	Font size of the menu bar text

Command	DISPlay:FSIGn
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable showing the result of of pass / fail test

Command	DISPlay:GRAPh:ACTivate
Scope	Graph
Summary	Select active graph

Command	DISPlay:GRAPh:CHANnel
Parameters	int
Range	1 to nchannels
Scope	Graph
Summary	Assigns graph <graph> to channel <channel>

Command	DISPlay:GRAPh:LABel
Parameters	string
Scope	Graph
Summary	Show or hide the graph's y axis label

Command	DISPlay:GRAPh:TITLe:DATA
Parameters	string
Default	""
Scope	Graph
Summary	Set the graph's title

Command	DISPlay:GRAPh:TITLe[:STATe]
Parameters	bool
Default	0
Scope	Graph
Summary	Show or hide the graph's title

Command	DISPlay:GRAPh:TRACes
Parameters	vector<int>
Summary	Assigns graph <graph> and the specified list of traces to channel <channel>

Command	DISPlay:GRAPh:X:SPACing
Parameters	enum
ENum Values	LINear|OBASe
Default	OBASe
Scope	Graph
Summary	Set the x axis type (linear frequency / point order base)

Command	DISPlay:GRAPh:Y[:SCALe]:DIVisions
Parameters	int
Range	4 to 30
Default	10
Scope	Graph
Summary	Set the number of y axis divisions

Command	DISPlay:IMAGe
Parameters	enum
ENum Values	NORMal|INVert
Default	NORMal
Scope	Instrument
Summary	Normal or inverted color scheme

Command	DISPlay:MAXimize
Parameters	bool
Default	0
Scope	Instrument
Summary	Enable / disable maximization of the active channel

Command	DISPlay:MENU[:STATe]
Parameters	bool
Default	1
Scope	Instrument
Summary	Show / hide the menu bar

Command	DISPlay:SKEY[:STATe]
Parameters	bool
Default	1
Scope	Instrument
Summary	Show / hide soft key labels

Command	DISPlay:TABLe:TYPE
Parameters	enum
ENum Values	MARKer|LIMit|SEGMent|ECHO|PLOSs|SCFactor|RLIMit
Default	MARKer
Scope	Instrument
Summary	Configure the table window (marker table / limit table / segment table / echo window / loss compensation table / power sensor calibration factor table / ripple test table)

Command	DISPlay:TABLe[:STATe]
Parameters	bool
Default	0
Scope	Instrument
Summary	Enable / disable display of the table window selected by :DISP:TABLe:TYPE

Command	DISPlay:UPDate
Summary	One-time update of the display when :DISP:ENAB is off

6.13. Data Operations (Read / Write; Import / Export)

6.13.1. INSTRUMENT STATES

Instrument settings, calibration state and measurement data can all be saved to a file on the host PC. All data associated with a measurement can be exported, including calibration data, match networks, measurement data, parameters, setup, and graphs, with zoom and marker information.

Command	`MMEMory:LOAD[:STATe]`
Parameters	string
Scope	Instrument
Summary	Recall the instrument state (settings, calibration & data) from the specified saved filename.
Example	`MMEMory:LOAD:STATe "C:/Users/test/Desktop/evna.cdstate"`

Command	`MMEMory:STORe:SALL`
Parameters	bool
Default	0
Scope	Instrument
Summary	Specify which channels and traces to include when saving an instrument state file: 1 = All channels and traces 0 = Only active channels and traces
Example	`MMEMory:STORe:SALL 1` `MMEMory:STORe:SALL? ==> 1`

Command	`MMEMory:STORe:STYPe`
Parameters	enum
ENum Values	STATe|CSTate|DSTate|CDSTate
Default	CSTate
Scope	Instrument
Summary	Specify the type of data to include when saving an instrument state file: STATe = measurement conditions only CSTate = measurement conditions + calibration state DSTate = measurement conditions + trace data CDSTate = measurement conditions + cal state + trace data
Example	`MMEMory:STORe:STYPe CDState` `MMEMory:STORe:STYPe? ==> CDSTate`

Command	`MMEMory:STORe[:STATe]`
Parameters	string
Scope	Instrument
Summary	Store the instrument state to the specified filename
Example	`MMEMory:STORe:STATe "C:/Users/test/Desktop/evna.cdstate"`

6.13.2. S-PARAMETERS (TOUCHSTONE FORMAT)

S-parameter measurement data can be exported in Touchstone format as an s1p or s2p file

Command	MMEMory:STORe:SNP:FORMat
Parameters	enum
ENum Values	AUTO|MA|DB|RI
Scope	Instrument
Summary	Specify the data format for the s-parameter touchstone file: AUTO = display format MA = linear magnitude + angle DB = logarithmic magnitude + angle RI = real + imaginary

Command	MMEMory:STORe:SNP:NPORTS
Parameters	int
Range	1 to nPorts
Default	1
Scope	Instrument
Summary	Number of ports used for the s-parameter data

Command	MMEMory:STORe:SNP:PORTS
Parameters	int,...
Range	1 to nPorts
Default	1,2,...
Scope	Instrument
Summary	List of the specific ports used for the s-parameter data

Command	MMEMory:STORe:SNP:TYPE:S1P
Parameters	int
Default	1
Summary	Specify the port to be used for s1p data

Command	MMEMory:STORe:SNP:TYPE:S2P
Parameters	int,int
Default	1,2
Summary	Specify the ports to be used for s2p data

Command	MMEMory:STORe:SNP[:DATA]
Parameters	string
Scope	Instrument
Summary	Save the active channel s-parameter data in touchstone format with the specified filename
Summary	MMEMory:STORe:SNP:DATA "C:/Users/test/Desktop/evna_data.s2p"

6.13.3. DATA ARRAYS

Retrieve or set the trace data as an array of raw, corrected or formatted measurement points.

Command	FORMat:BORDer
Parameters	enum
ENum Values	NORMal|SWAPped
Default	NORMal
Scope	Instrument
Summary	Specify the byte transfer order in binary format: Normal = MSB first Swapped = LSB first
Example	FORMAT:BORDer SWAPped FORMAT:BORDer? ==> SWAPped

Command	FORMat:DATA
Parameters	enum
ENum Values	ASCii|REAL|REAL32
Default	ASCii
Scope	Instrument
Summary	Set the format for data returns from the eVNA between ASCII and real (64-bit binary / 32-bit binary). Real formats are useful when the ASCII formatted responses do not contain sufficient precision in the measured data values.
Example	FORMat:DATA ASCii FORMat:DATA? ==> ASC

Command	CALCulate<ch>[:SELected]:DATA:FDATa
Parameters	double,…
Range	YMin,YMin to YMax,YMax
Scope	Trace
Summary	Read or set the array of formatted measurement data for the active trace on the specified channel, corrected and formatted per the trace settings. Two data values are returned for each point on the measurement trace to allow for primary and secondary values in Smith / polar formats. The second data value for each measurement point will be 0 for all other formats. array[1] = point 1 primary value array[2] = point 1 secondary value … array[2n-1] = point n primary value array[2n] = point n secondary value
Example	CALCulate:SELected:DATA:FDATa -59.9229,0,-68.6565,0,-56.3063,0, -60.4493,0,-56.8599,0,-54.1527,0,-60.8429,0 ASCII formatted response: FORMat:DATA ASCii CALCulate:SELected:DATA:FDATa? ==> -0.126978,0, -0.127947,0,-0.126675,0, -0.127523,0,-0.126646,0, -0.126373,0,-0.125918,0 REAL32 formatted response (array of bytes): FORMat:DATA REAL32 CALCulate:SELected:DATA:FDATa? ==> [35, 54, 48, 48, 48, 48, 53, 54, 190, 1, 184, 193, 0, 0, 0, 0, 190, 2, 204, 17, 0, 0, 0, 0, 190, 2, 134, 110, 0, 0, 0, 0, 190, 2, 72, 61, 0, 0, 0, 0, 190, 1, 72, 61, 0, 0, 0, 0, 190, 2, 69, 79, 0, 0, 0, 0, 190, 1, 168, 76, 0, 0, 0, 0, 10] REAL32 formatted response (after additional post-processing to convert bytes into real values): FORMat:DATA REAL32 CALCulate:SELected:DATA:FDATa? ==> [-0.1266813427209854, 0.0, -0.1277315765619278, 0.0, -0.12746593356132507, 0.0, -0.12722869217395782, 0.0, -0.12625212967395782, 0.0, -0.12721751630306244, 0.0, -0.12661856412887573, 0.0]

Command	CALCulate<ch>[:SELected]:DATA:FMEMory
Parameters	double,…
Range	YMin,YMin to YMax,YMax
Scope	Trace
Summary	Read the array of formatted data from memory for the active trace on the specified channel. Two data values are returned for each point on the measurement trace to allow for primary and secondary values in Smith / polar formats. The second data value for each measurement point will be 0 for all other formats. array[1] = point 1 primary value array[2] = point 1 secondary value … array[2n-1] = point n primary value array[2n] = point n secondary value
Example	CALCulate:SELected:DATA:FMEMory -10,0,-10,0,-10,0,-10,0,-10,0,-10,0, -10,0 CALCulate:SELected:DATA:FMEMory? ==> -10,0,-10,0,-10,0,-10,0,-10,0, -10,0,-10,0

Command	CALCulate<ch>[:SELected]:DATA:RDATa
Parameters	double,…
Range	YMin,YMin to YMax,YMax
Scope	Trace
Summary	Read or set the array of raw data for the active trace on the specified channel, uncorrected and unformatted. Two data values are returned for each point on the measurement trace. array[1] = point 1 real value array[2] = point 1 imaginary value … array[2n-1] = point n real value array[2n] = point n imaginary value
Example	CALCulate:SELected:DATA:RDATa 0.000755372,0.000668244,0.000166974, 0.000329551,-2.30479e-05,-0.00152987,-0.000350634, -0.000882104,-0.00142575,-0.000169225,0.00115221, -0.0015869,-0.000880961,0.000218272 ASCII formatted response: FORMat:DATA ASCii CALCulate:SELected:DATA:RDATa? ==> 0.32958,0.928687, 0.329659,0.928773,0.329532,0.9287, 0.329555,0.928815,0.329572,0.928609, 0.329693,0.928736,0.329582,0.9287 REAL32 formatted response (array of bytes): FORMat:DATA REAL32 CALCulate:SELected:DATA:RDATa? ==> [35, 54, 48, 48, 48, 48, 53, 54, 62, 168, 199, 243, 63, 109, 201, 71, 62, 168, 182, 81, 63, 109, 196, 8, 62, 168, 165, 157, 63, 109, 188, 120, 62, 168, 195, 85, 63, 109, 197, 252, 62, 168, 196, 115, 63, 109, 187, 12, 62, 168, 201, 125, 63, 109, 192, 198, 62, 168, 215, 241, 63, 109, 199, 92, 10] REAL32 formatted response (after additional post-processing to convert bytes into real values): FORMat:DATA REAL32 CALCulate:SELected:DATA:RDATa? ==> [0.32965049147605896, 0.9288524985313416, 0.3295159637928009, 0.9287724494934082, 0.3293885290622711, 0.928657054901123, 0.3296152651309967, 0.9288022518157959, 0.3296237885951996, 0.9286353588104248, 0.3296622335910797, 0.9287227392196655, 0.3297725021839142, 0.9288232326507568]

Command	CALCulate<ch>[:SELected]:DATA:SDATa
Parameters	double,…
Range	YMin,YMin to YMax,YMax
Scope	Trace
Summary	Read or set the array of complex corrected measurement data for the active trace on the specified channel, corrected but unformatted. Two data values are returned for each point on the measurement trace. array[1] = point 1 real value array[2] = point 1 imaginary value … array[2n-1] = point n real value array[2n] = point n imaginary value
Example	CALCulate:SELected:DATA:SDATa 0.000752236,0.000668967,0.000166862, 0.00032973,-2.2865e-05,-0.00152912,-0.00035028, -0.000882487,-0.00142602,-0.000168519,0.00115259, -0.00158577,-0.000880764,0.000217723 ASCII formatted response: FORMat:DATA ASCii CALCulate:SELected:DATA:SDATa? ==> 0.329579,0.928743, 0.329419,0.928683,0.329716,0.928731, 0.329482,0.928711,0.32968,0.928747, 0.329725,0.928764,0.329688,0.928832 REAL32 formatted response (array of bytes): FORMat:DATA REAL32 CALCulate:SELected:DATA:SDATa? ==> [35, 54, 48, 48, 48, 48, 53, 54, 62, 168, 172, 90, 63, 109, 197, 167, 62, 168, 178, 133, 63, 109, 191, 28, 62, 168, 200, 125, 63, 109, 201, 2, 62, 168, 202, 192, 63, 109, 186, 234, 62, 168, 206, 207, 63, 109, 182, 227, 62, 168, 205, 158, 63, 109, 189, 127, 62, 168, 195, 249, 63, 109, 188, 239, 10] REAL32 formatted response (after additional post-processing to convert bytes into real values): FORMat:DATA REAL32 CALCulate:SELected:DATA:SDATa? ==> [0.3294399380683899, 0.92879718542099, 0.32948699593544006, 0.9286973476409912, 0.32965460419654846, 0.928848385810852, 0.32967185974121094, 0.9286333322525024, 0.32970282435417175, 0.928571879863739, 0.3296937346458435, 0.928672730922699, 0.3296201527118683, 0.9286641478538513]

Command	CALCulate<ch>[:SELected]:DATA:SMEMory
Parameters	double,...
Range	YMin,YMin to YMax,YMax
Scope	Trace
Summary	Read or set the array of complex corrected data from memory for the active trace on the specified channel. Two data values are returned for each point on the measurement trace. array[1] = point 1 real value array[2] = point 1 imaginary value ... array[2n-1] = point n real value array[2n] = point n imaginary value
Example	CALCulate:SELected:DATA:SMEMory -0.997196,0.0248926,0.330012,0.928627, 0.896212,-0.405624,0.396256,-0.910373,-0.667643, -0.734523,-0.971169,0.161981,-0.433592,0.876005 CALCulate:SELected:DATA:SMEMory? ==> -0.997196,0.0248926,0.330012, 0.928627,0.896212,-0.405624,0.396256,-0.910373,-0.667643, -0.734523,-0.971169,0.161981,-0.433592,0.876005

6.13.4. DATA / MEMORY OPERATIONS (MEMORY MATH)

Work with trace data in memory and configure mathematic operations between the live and stored trace data.

Command	CALCulate[:SELected]:MATH:FUNCtion
Parameters	enum
ENum Values	NORMal|SUBTract|DIVide|ADD|MULTiply
Default	NORMal
Scope	Trace
Summary	Perform a mathematic operation on the active trace between the live measurement data and the data stored in memory: Normal = show data only Subtract = data - memory Divide = data / memory Add = data + memory Multiply = data * memory

Command	CALCulate[:SELected]:MATH:MEMorize
Scope	Trace
Summary	Copy the active trace measurement data to memory

Command	DISPlay:CHANnel:TRACe:MEMory[:STATe]
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable display of the trace stored in memory

6.14. Simulator Mode

Operate the eVNA in simulator mode to allow demonstration of the software and features without the eVNA hardware or DUT

Command	SIMulator:FILE:FET
Parameters	int, string
Summary	Load the factory error terms file for the given port from CSV (port, filename)

Command	SIMulator:FILE:FOLDer
Parameters	string
Summary	Specify the folder where the similualtor files can be found

Command	SIMulator:FILE:LCOM
Parameters	string
Summary	Load the simulator loss compensation file (LCF)

Command	SIMulator:FILE:PCF
Parameters	string
Summary	Load the simulator PCF file

Command	SIMulator:FILE:RTF
Parameters	string
Summary	Load the simulator RTF file

Command	SIMulator:FILE:STF
Parameters	string
Summary	Load the simulator STF file

Command	SIMulator:FILE:UET
Parameters	string
Summary	Load the user error terms file from CSV

Command	SIMulator:FILEname
Parameters	string
Default	""
Summary	Load the Touchstone file describing the DUT

Command	SIMulator:NF
Parameters	double
Range	-200 to 0
Default	-120
Units	dBFS/Hz
Summary	Set the noise floor for the simulator

Command	SIMulator:RCF
Parameters	double
Range	1 to 150
Default	100
Summary	Set the RCF value for the simulator

6.15. Calibration

Calibration and error correction configuration

Command	SENSe:CORRection:CLEar
Scope	Channel
Summary	Clear calibration coefficients

Command	SENSe:CORRection:COEFficient[:DATA]
Parameters	in: enum, int, int in/out: double, …
ENum Values	ES|ER|ED|EL|ET|EX
Scope	Channel
Summary	Set the calibration coefficients for the active channel. 4 parameters are required (string, int, int, array): 1) Coefficient type: ES (source match), ER (reflection tracking), ED (directivity), EL (load match), ET (transmission tracking), or EX (isolation) 2) Response port (1 or 2) 3) Stimulus port (1 or 2) 4) Array with 2 elements per measurement point (real and imaginary coefficients)

Command	SENSe:CORRection:STATe
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable error correction for the active channel

Command	SENSe:CORRection:TRIGger:FREE
Parameters	bool
Default	1
Scope	Channel
Summary	Enable / disable the internal (free running) trigger source for calibrations on the active channel

Command	SENSe:CORRection:TYPE?
Parameters	enum
ENum Values	ERES|NONE|RESPO|RESPS|RESPT|SOLT|SMIX
Scope	Trace
Summary	Read the calibration type for the active channel: ERES = enhanced response NONE = no calibration RESPO = open response RESPS = short response RESPT = thru response SOLT = SOLT calibration

Command	SYSTem:CORRection:PERFormance
Parameters	bool
Default	0
Scope	Instrument
Summary	Enable / disable corrected performance measurements

Command	SYSTem:CORRection[:STATe]
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable system error correction

6.15.1. CALIBRATION TYPE

There are a number of calibration methods possible, the choice of which is dictated by the goals of the measurement session including the required accuracy and the type of device to be measured.

Command	SENSe:CORRection:COEFficient:METHod:ERESponse
Parameters	int, int
Range	1 to nPorts
Scope	Channel
Summary	Set enhanced response calibration for the active channel, specifying the response and stimulus ports (response port, stimulus port)

Command	SENSe:CORRection:COEFficient:METHod:SOLT
Parameters	int

Range	1 to nPorts
Scope	Channel
Summary	Set full n port calibration method

Command	SENSe:CORRection:COEFficient:METHod:SOLT:NPORTS
Parameters	int
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Set the number of ports for SOLT calibration

Command	SENSe:CORRection:COEFficient:METHod:SOLT:PORTS
Parameters	int,...
Range	1 to nPorts
Default	1,2,...
Scope	Channel
Summary	Set the ports to be used for SOLT calibration

Command	SENSe:CORRection:COEFficient:METHod:SOLT1
Parameters	int
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Set the port for 1-port SOL calibration

Command	SENSe:CORRection:COEFficient:METHod:SOLT2
Parameters	int,int
Range	1 to nPorts
Default	1,2
Scope	Channel
Summary	Set the ports for 2-port SOL calibration

Command	SENSe:CORRection:COEFficient:METHod[:RESPonse]:OPEN
Parameters	int
Range	1 to nPorts
Scope	Channel
Summary	Set the calibration method on the response port to the open standard

Command	SENSe:CORRection:COEFficient:METHod[:RESPonse]:SHORt
Parameters	int
Range	1 to nPorts
Scope	Channel
Summary	Set the calibration method on the response port to the short standard

Command	SENSe:CORRection:COEFficient:METHod[:RESPonse]:THRU
Parameters	int, int
Range	1 to nPorts
Scope	Channel

Summary	Set the calibration method to the thru standard, specifying the port order (response port, stimulus port)

Command	SENSe:CORRection:COEFficient:SAVE
Summary	Save and apply the relevant calibration coefficients based on the previous settings

Command	SENSe:CORRection:COLLect:METHod:ERESponse
Parameters	int, int
Range	1 to nPorts
Scope	Channel
Summary	Select enhanced response as the calibration method and specify the port order {response port, stimulus port}

Command	SENSe:CORRection:COLLect:METHod:SOLT
Scope	Channel
Summary	Select n-port SOL as the calibration method

Command	SENSe:CORRection:COLLect:METHod:SOLT:NPORTS
Parameters	int
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Set the number of ports for the SOLT calibration

Command	SENSe:CORRection:COLLect:METHod:SOLT:PORTS
Parameters	int,…
Range	1 to nPorts
Default	1,2,…
Scope	Channel
Summary	Set the ports and port order to be used for SOLT calibration

Command	SENSe:CORRection:COLLect:METHod:SOLT1
Parameters	int
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Set 1-port SOL calibration and specify the port number

Command	SENSe:CORRection:COLLect:METHod:SOLT2
Parameters	int,int
Range	1 to nPorts
Default	1,2
Scope	Channel
Summary	Set 2-port SOL calibration and specify the port order

Command	SENSe:CORRection:COLLect:METHod[:RESPonse]:OPEN
Parameters	int
Range	1 to nPorts
Scope	Channel
Summary	Select open response as the calibration method for the given port

Command	SENSe:CORRection:COLLect:METHod[:RESPonse]:SHORt
Parameters	int
Range	1 to nPorts
Scope	Channel
Summary	Select short response as the calibration method for the given port

Command	SENSe:CORRection:COLLect:METHod[:RESPonse]:THRU
Parameters	int, int
Range	1 to nPorts
Scope	Channel
Summary	Select thru response as the calibration method and specify the port order {port1, port2}

Command	SENSe:CORRection:COLLect:PARTial:SAVE
Scope	Channel
Summary	Recalculate calibration coefficients taking into account partially overridden data

Command	SENSe:CORRection:COLLect:SAVE
Scope	Channel
Summary	Calculate all calibration coefficients based on the previously entered parameters

6.15.2. CALIBRATION MEASUREMENT

Measure the required calibration standards

Command	SENSe:CORRection:COLLect:CLEar
Scope	Channel
Summary	Clear measurement data for the mechanical cal kit on the active channel

Command	SENSe:CORRection:COLLect[:ACQuire]:ISOLation
Parameters	int, int
Range	1 to nPorts
Scope	Channel
Summary	Measure isolation between ports, specifying the port order (receiver port, source port)
Command	SENSe:CORRection:COLLect[:ACQuire]:LOAD
Parameters	int

Range	1 to nPorts
Scope	Channel
Summary	Measure the load standard on the specified source port

Command	SENSe:CORRection:COLLect[:ACQuire]:OPEN
Parameters	int
Range	1 to nPorts
Scope	Channel
Summary	Measure the open standard on the specified source port

Command	SENSe:CORRection:COLLect[:ACQuire]:SHORt
Parameters	int
Range	1 to nPorts
Scope	Channel
Summary	Measure the short standard on the specified source port

Command	SENSe:CORRection:COLLect[:ACQuire]:SUBClass
Parameters	int
Range	1 to nSubClasses
Scope	Channel
Summary	Specify the subclass used for calibration

Command	SENSe:CORRection:COLLect[:ACQuire]:THRU
Parameters	int, int
Range	1 to nPorts
Scope	Channel
Summary	Measure the thru standard, specifying the port order (receiver port, source port)

6.15.3. CALIBRATION KITS

Define / modify calibration kits

Command	MMEMory:LOAD:CKIT
Parameters	[int], string
Range	1 to nCalKits
Default	1
Scope	Instrument
Summary	Load the calibration kit definition from .csv file {cal kit no., filename}

Command	MMEMory:STORe:CKIT
Parameters	[int], string
Range	1 to nCalKits
Default	1
Scope	Instrument
Summary	Store the calibration kit definition in .csv file {cal kit no., filename}
Command	SENSe:CORRection:COLLect:CKIT
Parameters	[int]
Summary	Specify calibration kit number

Command	SENSe:CORRection:COLLect:CKIT:LABel
Parameters	string
Default	A set of calibration kit part numbers with built-in software support
Scope	Cal kit
Summary	Set a label / name for the active cal kit

Command	SENSe:CORRection:COLLect:CKIT:ORDer:LABel
Parameters	string
Summary	Set a subclass Label for the active cal kit

Command	SENSe:CORRection:COLLect:CKIT:ORDer:LOAD
Parameters	int, int
Range	1, 1 to nPorts, nStandards
Scope	Channel
Summary	Set the load standard used for a specific port {port, standard number}

Command	SENSe:CORRection:COLLect:CKIT:ORDer:OPEN
Parameters	int, int
Range	1, 1 to nPorts, nStandards
Scope	Channel
Summary	Set the open standard used for a specific port {port, standard number}

Command	SENSe:CORRection:COLLect:CKIT:ORDer:SHORt
Parameters	int, int
Range	1, 1 to nPorts, nStandards
Scope	Channel
Summary	Set the short standard used for a specific port {port, standard number}

Command	SENSe:CORRection:COLLect:CKIT:ORDer:THRU
Parameters	int, int, int
Range	1,1,1 to nPorts, nPorts, nStandards
Scope	Channel
Summary	Set the thru standard used, along with the port order {port1, port2, standard number}

Command	SENSe:CORRection:COLLect:CKIT:ORDer[:SELect]
Parameters	int
Range	1 to nSubClasses
Scope	Channel
Summary	Set the calibration standard sub-class (frequency range ID)

Command	SENSe:CORRection:COLLect:CKIT:RESet
Scope	Channel
Summary	Reset the calibration kit selection to a preset value for the active cal kit

Command	SENSe:CORRection:COLLect:CKIT:STAN
Parameters	[int]
Range	1 to nStandards
Default	1
Scope	Cal kit
Summary	Set the standard number for the active cal kit

Command	SENSe:CORRection:COLLect:CKIT:STAN:ARBitrary
Parameters	double
Range	-1E+18 to 1E+18
Default	50
Units	Ohm
Scope	Cal kit, standard
Summary	Set the impedance value for the arbitary impedance standard of the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:C0
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	fF
Scope	Cal kit, standard
Summary	Set the C0 capacitance value for the for the open cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:C1
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	1E-27 F/Hz
Scope	Cal kit, standard
Summary	Set the C1 capacitance value for the for the open cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:C2
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	1E-36 F/Hz^2
Scope	Cal kit, standard
Summary	Set the C2 capacitance value for the for the open cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:C3
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	1E-45 F/Hz^3
Scope	Cal kit, standard
Summary	Set the C3 capacitance value for the for the open cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:CHARacter
Parameters	enum
ENum Values	COAXial|WAVeguide
Default	COAXial
Scope	Cal kit, standard
Summary	Set the media type for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:DELay
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	sec
Scope	Cal kit, standard
Summary	Set the offset delay for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:FILE
Parameters	string
Default	""
Scope	Cal kit, standard
Summary	Set the Touchstone filename for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:FMAXimum
Parameters	double
Range	0 to 1000000000000
Default	0
Units	Hz
Scope	Cal kit, standard
Summary	Set the maximum frequency for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:FMINimum
Parameters	double
Range	0 to 1000000000000
Default	9.99E+11
Units	Hz
Scope	Cal kit, standard
Summary	Set the minimum frequency for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:L0
Parameters	double

Range	-1E+18 to 1E+18
Default	0
Units	pH
Scope	Cal kit, standard
Summary	Set the L0 impedance value for the for the short cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:L1
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	1E-24 H/Hz
Scope	Cal kit, standard
Summary	Set the L1 impedance value for the for the short cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:L2
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	1E-33 H/Hz^2
Scope	Cal kit, standard
Summary	Set the L2 impedance value for the for the short cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:L3
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	1E-42 H/Hz^3
Scope	Cal kit, standard
Summary	Set the L3 impedance value for the for the short cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:LABel
Parameters	string
Scope	Cal kit, standard
Summary	Set a label / name for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:LOSS
Parameters	double
Range	-1E+18 to 1E+18
Default	0
Units	Ohm/sec
Scope	Cal kit, standard
Summary	Set the offset loss for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT:STAN:TYPE
Parameters	enum
ENum Values	OPEN|SHORt|LOAD|THRU|UTHR|ARBI|NONE
Scope	Cal kit, standard

Summary	Specify the standard type for the active cal standard (open / short / load / thru / unknown thru / arbitrary impedance / none = ideal)

Command	SENSe:CORRection:COLLect:CKIT:STAN:Z0
Parameters	double
Range	-1E+18 to 1E+18
Default	50
Units	Ohm
Scope	Cal kit, standard
Summary	Set the characteristic impedance for the active cal standard

Command	SENSe:CORRection:COLLect:CKIT[:SELect]
Range	1 to nCalKits
Default	1
Scope	Channel
Summary	Set the calibration kit index number for the active channel

6.15.4. POWER CALIBRATION

The eVNA supports power calibration of ports 1 and 2 independently with an external power meter

Command	MMEMory:LOAD:ASCFactor
Parameters	[int], string
Scope	Instrument
Summary	Load the power sensor calibration factor table from .csv file {port, filename}

Command	MMEMory:LOAD:PROSs
Parameters	[int], string
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Load the loss compensation table for the specified port from .csv file {port, filename}

Command	MMEMory:STORe:PLOSs
Parameters	[int], string
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Store the loss compensation table for the specified port in .csv file {port, filename}

Command	MMEMory:STORe:PSCFactor
Parameters	[int], string
Scope	Instrument
Summary	Store the power sensor calibration factor table in .csv file {port, filename}

Command	SOURce:POWer:PORT:CORRection:COLLect:ASENsor:RCFactor
Parameters	double
Range	1 to 150
Default	100
Units	%
Scope	Instrument
Summary	Set the reference calibration factor at 50MHz for the power sensor

Command	SOURce:POWer:PORT:CORRection:COLLect:AVERage[:COUNt]
Parameters	int
Range	1 to 100
Default	1
Scope	Channel, port
Summary	Set the averaging count for power calibration measurements

Command	SOURce:POWer:PORT:CORRection:COLLect:NTOLerance
Parameters	double
Range	0 to 100
Default	5
Units	dB
Scope	Channel, port
Summary	Set the tolerance of the power calibration data

Command	SOURce:POWer:PORT:CORRection:COLLect:TABLe:ASENsor:DATA
Parameters	double,…
Scope	Instrument
Summary	Set the calibration table array for power sensor: array[0] = Number of calibration points array[1] = Frequency of point 1 (1000 - 500e9) array[2] = Calibration factor for point 1 (1% to 150%) … array[2n-1] = Frequency of nth point (1000 - 500e9) array[2n] = Calibration factor for nth point (1% to 150%)

Command	SOURce:POWer:PORT:CORRection:COLLect:TABLe:LOSS:DATA
Parameters	double,...
Scope	Channel, port
Summary	Set the loss compensation table for the power meter: array[0] = Number of calibration points array[1] = Frequency of point 1 (1000 - 500e9) array[2] = Loss for point 1 (1% to 150%) ... array[2n-1] = Frequency of nth point (1000 - 500e9) array[2n] = Loss for nth point (1% to 150%)

Command	SOURce:POWer:PORT:CORRection:COLLect:TABLe:LOSS[:STATe]
Parameters	bool
Summary	Enable / disable loss compensation

Command	SOURce:POWer:PORT:CORRection:COLLect[:ACQuire]
Parameters	{ASENsor}
Scope	Channel, port
Summary	Acquire power calibration data from power sensor "ASensor"

Command	SOURce:POWer:PORT:CORRection:DATA
Parameters	double,...
Summary	Set the power correction data array

Command	SOURce:POWer:PORT:CORRection:REFdata
Parameters	double,...
Summary	Set the reference correction data array

Command	SOURce:POWer:PORT:CORRection[:STATe]
Parameters	bool
Default	0
Scope	Channel, port
Summary	Enable / disable power level error correction

Command	SYSTem:COMMunicate:VISA:PMETer:ADDRess
Parameters	string
Default	""
Scope	Instrument
Summary	Set the VISA address of the power meter accessory

Command	SYSTem:COMMunicate:VISA:PMETer:DISCover?
Summary	Provide a list of available power meters on the USB bus

Command	SYSTem:COMMunicate:VISA:PMETer:MODel
Parameters	enum
ENum Values	RNS_NRP_Z|RNS_NRP_XXS_SN
Scope	Instrument
Summary	Set the model name of the power meter accessory

6.15.5. RECEIVER CALIBRATION

A receiver calibration can be performed on ports 1 and 2 for measurements requiring an accurate output power level.

Command	SENSe:CORRection:RECeiver
Parameters	[int]
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Set the receiver port number

Command	SENSe:CORRection:RECeiver:COLLect:ACQuire
Parameters	int
Range	1 to nPorts
Default	1
Scope	Channel
Summary	Acquire receiver calibration for receiver port when the specified source port is transmitting

Command	SENSe:CORRection:RECeiver[:STATe]
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable receiver correction

6.15.6. PORT EXTENSIONS

Port extension is a allows the reference plane to be moved from the calibration plane to the DUT edges by adjusting for time delay and loss in the test fixturing.

Command	SENSe:CORRection:EXTension:PORT
Parameters	[int]
Range	1 to nPorts
Default	1
Units	none, sec
Scope	Channel
Summary	Set the delay offset for the port extension on the specified port {[port,] delay}

Command	SENSe:CORRection:EXTension:PORT:FREQuency
Parameters	[int], double
Range	1, fMin to 2, fMax
Default	1, 1e9
Units	none, Hz
Scope	Channel
Summary	Set the frequency value at which loss 1 or loss 2 applies {loss number (1 or 2), frequency}

Command	SENSe:CORRection:EXTension:PORT:INCLude[:STATe]
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable port extension loss correction on the active port

Command	SENSe:CORRection:EXTension:PORT:LDC
Parameters	double
Range	-90 to 90
Default	0
Units	dB
Scope	Channel
Summary	Set the port extension loss value to be applied at DC

Command	SENSe:CORRection:EXTension:PORT:LOSS
Parameters	[int], double
Range	1, -90 to 2, 90
Default	1, 0
Units	none, dB
Scope	Channel
Summary	Set the port extension loss value to be applied for loss 1 or loss 2 at the specified frequencies. Where only loss 1 is set the loss value calculation is purely a function of frequency. Where loss 1 and loss 2 are set, the loss values are applied at the 2 frequencies specified and extrapolated. {loss number (1 or 2), loss}

Command	SENSe:CORRection:EXTension[:STATe]
Parameters	bool
Default	0
Scope	Channel
Summary	Enable / disable port extension

6.16. Time Domain

Configure the time domain gating functions using either center and span times, or start and stop times

Command	CALCulate[:SELected]:FILTer[:GATE]:TIME:CENTer
Parameters	double
Default	0
Units	sec
Scope	Trace
Summary	Set the gate filter center time on the specified channel

Command	CALCulate[:SELected]:FILTer[:GATE]:TIME:SHAPe
Parameters	enum
ENum Values	MAXimum|WIDE|NORMal|MINimum
Default	NORMal
Scope	Trace
Summary	Set the gate filter shape from MIN (smallest possible gate filter) to MAX (widest possible gate filter)

Command	CALCulate[:SELected]:FILTer[:GATE]:TIME:SPAN
Parameters	double
Default	0.00000002
Units	sec
Scope	Trace
Summary	Set the gate filter time span on the specified channel

Command	CALCulate[:SELected]:FILTer[:GATE]:TIME:STARt
Parameters	double
Default	-0.00000001
Units	sec
Scope	Trace
Summary	Set the gate filter start time on the specified channel

Command	CALCulate[:SELected]:FILTer[:GATE]:TIME:STATe
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable time domain gating on the specified channel

Command	CALCulate[:SELected]:FILTer[:GATE]:TIME:STOP
Parameters	double
Default	0.00000001
Units	sec
Scope	Trace
Summary	Set the gate filter stop time on the specified channel
Command	CALCulate[:SELected]:FILTer[:GATE]:TIME[:TYPE]
Parameters	enum
ENum Values	BPASs|NOTCh

Default	BPASs
Scope	Trace
Summary	Set the type / shape of the gate filter on the specified channel. Band pass operates within the time range specified whereas notch excludes it.

Command	CALCulate[:SELected]:TRANsform:TIME:CENTer
Parameters	double
Default	0
Units	sec
Scope	Trace
Summary	Set the center time for time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:FIRSTnyquist
Summary	Set view to first Nyquist Zone - Start to 0 and Stop to Maximum Value

Command	CALCulate[:SELected]:TRANsform:TIME:IMPulse:WIDTh
Parameters	double
Default	Varies depending on the frequency span and the transformation type
Scope	Trace
Summary	Set the impulse response width for the time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:KBESsel
Parameters	double
Range	0 to 13
Default	6
Scope	Trace
Summary	Set the Kayser Bessel window width for the time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:LPFRequency
Summary	Set the frequency range to match the low pass filter time domain transform function

Command	CALCulate[:SELected]:TRANsform:TIME:SPAN
Parameters	double
Default	0.00000002
Units	sec
Scope	Trace
Summary	Set the time span for time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:STARt
Parameters	double
Default	-0.00000001
Units	sec
Scope	Trace
Summary	Set the start time for time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:STATe
Parameters	bool
Default	0
Scope	Trace
Summary	Enable / disable time domain transform
Command	CALCulate[:SELected]:TRANsform:TIME:STEP:RTIMe
Parameters	double
Default	Varies depending on the frequency span type
Scope	Trace
Summary	Sets the step rise time for time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:STIMulus
Parameters	enum
ENum Values	IMPulse|STEP
Default	IMPulse
Scope	Trace
Summary	Set the time domain transform stimulus type: step = step stimulus impulse = pulse stimulus

Command	CALCulate[:SELected]:TRANsform:TIME:STOP
Parameters	double
Default	0.00000001
Units	sec
Scope	Trace
Summary	Set the stop time for time domain transform

Command	CALCulate[:SELected]:TRANsform:TIME:UNITs
Parameters	enum
ENum Values	TIME|LENGth
Summary	Set the time domain display units

Command	CALCulate[:SELected]:TRANsform:TIME[:TYPE]
Parameters	enum
ENum Values	BPASs|LPASs
Default	BPASs
Scope	Trace
Summary	Set the time domain transform type (bandpass / lowpass)

6.17. System Configuration

Connection and configuration options

6.17.1. WARNING / ERROR OPTIONS

Audible notifications and error logging options

Command	SYSTem:BEEPer:COMPlete:IMMediate
Scope	Instrument
Summary	Generates a beep immdeiately for notification on completion of an operation

Command	SYSTem:BEEPer:COMPlete:STATe
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable audible beeps for completion of an operation

Command	SYSTem:BEEPer:WARNing:IMMediate
Scope	Instrument
Summary	Generates a beep immediately for notification of warning / limit test results

Command	SYSTem:BEEPer:WARNing:STATe
Parameters	bool
Default	1
Scope	Instrument
Summary	Enable / disable audible beeps for system errors or warnings

Command	SYSTem:ERRor:ALL?
Parameters	{number, string, string},...
Scope	Instrument
Summary	Read all error / event information <error code>, <description>, >detail>; ...

Command	SYSTem:ERRor:CODE:ALL?
Parameters	int,...
Summary	Read all errors / event codes

Command	SYSTem:ERRor:CODE[:NEXT]?
Parameters	int
Summary	Read next error / event code

Command	SYSTem:ERRor:COUNt?
Parameters	int
Summary	Read number of errors / events

Command	SYSTem:ERRor[:NEXT]?
Parameters	number, string, string
Summary	Read next error / event information <error code>, <description>, >detail>

6.18. Status Registers

Review VNA status and events

6.18.1. BANDWIDTH LIMIT REGISTERS

Enable and query events based on bandwidth limit tests

Command	STATus:QUEStionable:BLIMit:CHANnel
Parameters	[int]
Range	1 to nChannels
Default	1
Summary	Specify the channel number for the Questionable Bandwidth Limit registers

Command	STATus:QUEStionable:BLIMit:CHANnel:CONDition?
Parameters	int
Scope	Channel
Summary	Read the Questionable Bandwidth Limit Channel Status Condition Register: Bits 0-13 = 0 (pass) or 1 (fail) for the bandwidth limit tests on traces 1-14 on the active channel

Command	STATus:QUEStionable:BLIMit:CHANnel:ECHannel:CONDition?
Parameters	int
Scope	Channel
Summary	Read the Questionable Bandwidth Limit Channel Extra Status Condition Register: Bits 14-15 = 0 (pass) or 1 (fail) for the bandwidth limit tests on traces 15-16 of the active channel

Command	STATus:QUEStionable:BLIMit:CHANnel:ECHannel:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Bandwidth Limit Channel Extra Status Enable Register

Command	STATus:QUEStionable:BLIMit:CHANnel:ECHannel:NTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Sets the negative transition filter of the Questionable Bandwidth Limit Channel Extra Status Event Register

Command	STATus:QUEStionable:BLIMit:CHANnel:ECHannel:PTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Sets the positive transition filter of the Questionable Bandwidth Limit Channel Extra Status Event Register

Command	STATus:QUEStionable:BLIMit:CHANnel:ECHannel[:EVENt]?
Parameters	int

Scope	Channel
Summary	Reads the Questionable Bandwidth Limit Channel Extra Status Event Register: Bits 14-15 = 0 (no event) or 1 (event) for traces 15-16 of the active channel

Command	STATus:QUEStionable:BLIMit:CHANnel:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Bandwidth Limit Channel Status Enable Register

Command	STATus:QUEStionable:BLIMit:CHANnel:NTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the negative transition of Questionable Bandwidth Limit Status Event Register

Command	STATus:QUEStionable:BLIMit:CHANnel:PTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the positive transition of Questionable Bandwidth Limit Status Event Register

Command	STATus:QUEStionable:BLIMit:CHANnel[:EVENt]?
Parameters	int
Scope	Channel
Summary	Read the Questionable Bandwidth Limit Status Event Register: Bits 0-1 = 0 (no event) or 1 (event) for traces 1-14

Command	STATus:QUEStionable:BLIMit:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Bandwidth Limit Status Condition Register: Bits 1-14 = 0 (pass) or 1 (fail) for channels 1-14 Bit 0 ORs the enabled bits of the extra status condition register

Command	STATus:QUEStionable:BLIMit:ELIMit:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Bandwidth Limit Extra Status Condition Register Bits 1-2 = 0 (pass) or 1 (fail) for channels 15-16 Bit 0 = 0

Command	STATus:QUEStionable:BLIMit:ELIMit:ENABle
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the Questionable Bandwidth Limit Extra Status Enable Register. Each bit enables the corresponding event bit to be propagated upwards

Command	STATus:QUEStionable:BLIMit:ELIMit:NTRansition
Parameters	int
Summary	Set the negative transition filter of the Questionable Bandwidth Limit Extra Status Register

Command	STATus:QUEStionable:BLIMit:ELIMit:PTRansition
Parameters	int
Summary	Set the positive transition filter of the Questionable Bandwidth Limit Extra Status Register

Command	STATus:QUEStionable:BLIMit:ELIMit[:EVENt]?
Parameters	int
Summary	Read the Questionable Bandwidth Limit Extra Status Event Register: Bits 1-2 = 0 (no event) or 1 (event) for channels 14-15

Command	STATus:QUEStionable:BLIMit:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Bandwidth Limit Status Enable Register. Each bit enables the corresponding trace event to be propagated to the channel event

Command	STATus:QUEStionable:BLIMit:NTRansition
Parameters	int
Summary	Set the negative transition filter of the Questionable Bandwidth Limit Status Register

Command	STATus:QUEStionable:BLIMit:PTRansition
Parameters	int
Summary	Set the positive transition filter of the Questionable Bandwidth Limit Status Register

Command	STATus:QUEStionable:BLIMit[:EVENt]?
Scope	Instrument
Summary	Read the Questionable Bandwidth Limit Status Event Register: Bits 1-14 = 0 (no event) or 1 (event) for channels 1-14 Bit 0 ORs the enabled bits of the Extra Status Event Register

6.18.2. LIMIT REGISTERS

Enable and query events based on limit tests

Command	STATus:QUEStionable:LIMit:CHANnel
Parameters	[int]
Range	1 to nChannels
Default	1
Summary	Specify the channel number

Command	STATus:QUEStionable:LIMit:CHANnel:CONDition?
Scope	Channel
Summary	Read the Questionable Limit Channel Status Condition Register: Bits 1-14 = 0 (pass) or 1 (fail) for traces 1-14 on the active channel

Command	STATus:QUEStionable:LIMit:CHANnel:ECHannel:CONDition?
Parameters	int
Scope	Channel
Summary	Read the Questionable Limit Channel Extra Status Condition Register: Bits 1-2 = 0 (pass) or 1 (fail) for traces 15-16 on the active channel

Command	STATus:QUEStionable:LIMit:CHANnel:ECHannel:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Limit Channel Extra Status Enable Register (for traces 15-16)

Command	STATus:QUEStionable:LIMit:CHANnel:ECHannel:NTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the negative transition filter of the Questionable Limit Channel Extra Status Register (for traces 15-16)

Command	STATus:QUEStionable:LIMit:CHANnel:ECHannel:PTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the positive transition filter of the Questionable Limit Channel Extra Status Register (for traces 15-16)

Command	STATus:QUEStionable:LIMit:CHANnel:ECHannel[:EVENt]?
Parameters	int
Scope	Channel
Summary	Read the Questionable Limit Channel Extra Status Event Register: Bits 1-2 = 0 (no event) or 1 (event) for traces 15-16 on the active channel

Command	STATus:QUEStionable:LIMit:CHANnel:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Limit Channel Status Enable Register (for traces 1-14)

Command	STATus:QUEStionable:LIMit:CHANnel:NTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the negative transition filter of the Questionable Limit Channel Status Register (for traces 1-14)

Command	STATus:QUEStionable:LIMit:CHANnel:PTRansition

Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the positive transition filter of the Questionable Limit Channel Status Register (for traces 1-14)

Command	STATus:QUEStionable:LIMit:CHANnel[:EVENt]?
Parameters	int
Scope	Channel
Summary	Read the Questionable Limit Channel Status Event Register: Bits 1-14 = 0 (no event) or 1 (event) for traces 1-14 on the active channel

Command	STATus:QUEStionable:LIMit:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Limit Status Condition Register: Bits 1-14 = 0 (pass) or 1 (fail) for traces 1-14 on the active channel

Command	STATus:QUEStionable:LIMit:ELIMit:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Limit Extra Status Condition Register: Bits 1-2 = 0 (pass) or 1 (fail) for traces 15-16 on the active channel

Command	STATus:QUEStionable:LIMit:ELIMit:ENABle
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the Questionable Limit Extra Status Enable Register (for traces 15-16)

Command	STATus:QUEStionable:LIMit:ELIMit:NTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the negative transition filter of Questionable Limit Extra Status Register

Command	STATus:QUEStionable:LIMit:ELIMit:PTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the positive transition filter of Questionable Limit Extra Status Register

Command	STATus:QUEStionable:LIMit:ELIMit[:EVENt]?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Limit Extra Status Event Register: Bits 1-2 = 0 (no event) or 1 (event) for traces 15-16 on the active channel

Command	STATus:QUEStionable:LIMit:ENABle
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the Questionable Limit Status Enable Register (for traces 1-14)

Command	STATus:QUEStionable:LIMit:NTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the negative transition filter of Questionable Limit Status Event Register

Command	STATus:QUEStionable:LIMit:PTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the positive transition filter of Questionable Limit Status Event Register

Command	STATus:QUEStionable:LIMit[:EVENt]?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Limit Status Event Register: Bits 1-14 = 0 (no event) or 1 (event) for traces 1-14 on the active channel

6.18.3. OPERATION REGISTERS

Command	STATus:OPERation:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Operation Status Condition Register

Command	STATus:OPERation:ENABle
Parameters	int
Range	0 to 65535
Default	0
Scope	Instrument
Summary	Set the operation Status Enable Register

Command	STATus:OPERation:NTRansition
Parameters	int
Range	0 to 65535
Default	0
Scope	Instrument
Summary	Sets the negative transition filter of the Operation Status Register. A bit set to 1 enables an event when the corresponding bit in the condition register changes from 1 to 0

Command	STATus:OPERation:PTRansition
Parameters	int
Range	0 to 65535
Default	0
Scope	Instrument
Summary	Sets the positive transition filter of the Operation Status Register. A bit set to 1 enables an event when the corresponding bit in the condition register changes from 0 to 1

Command	STATus:OPERation[:EVENt]?
Parameters	int
Scope	Instrument
Summary	Read the Operation Status Event Register

6.18.4. RIPPLE LIMIT REGISTERS

Enable and query events based on ripple limit tests

Command	STATus:QUEStionable:RLIMit:CHANnel
Parameters	[int]
Range	1 to nChannels
Default	1
Summary	Specify the channel number

Command	STATus:QUEStionable:RLIMit:CHANnel:CONDition?
Parameters	int

Scope	Channel
Summary	Read the Questionable Ripple Limit Channel Status Condition Register: Bits 1-14 = 0 (pass) or 1 (fail) for traces 1-14 on the active channel

Command	STATus:QUEStionable:RLIMit:CHANnel:ECHannel:CONDition?
Parameters	int
Scope	Channel
Summary	Read the Questionable Ripple Limit Channel Extra Status Condition Register: Bits 1-2 = 0 (pass) or 1 (fail) for traces 15-16 on the active channel

Command	STATus:QUEStionable:RLIMit:CHANnel:ECHannel:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Ripple Limit Channel Extra Status Enable Register (for traces 15-16)

Command	STATus:QUEStionable:RLIMit:CHANnel:ECHannel:NTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the negative transition filter of the Questionable Ripple Limit Channel Extra Status Register

Command	STATus:QUEStionable:RLIMit:CHANnel:ECHannel:PTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the positive transition filter of the Questionable Ripple Limit Channel Extra Status Register

Command	STATus:QUEStionable:RLIMit:CHANnel:ECHannel[:EVENt]?
Parameters	int
Scope	Channel
Summary	Read the Questionable Ripple Limit Channel Extra Status Event Register: Bits 1-2 = 0 (no event) or 1 (event) for traces 15-16 on the active channel

Command	STATus:QUEStionable:RLIMit:CHANnel:ENABle
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the Questionable Ripple Limit Channel Status Enable Register (for traces 1-14)

Command	STATus:QUEStionable:RLIMit:CHANnel:NTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the negative transition filter of the Questionable Ripple Limit Channel Status Register

Command	STATus:QUEStionable:RLIMit:CHANnel:PTRansition
Parameters	int
Range	0 to 65535
Scope	Channel
Summary	Set the positive transition filter of the Questionable Ripple Limit Channel Status Register

Command	STATus:QUEStionable:RLIMit:CHANnel[:EVENt]?
Parameters	int
Scope	Channel
Summary	Reads the Questionable Ripple Limit Channel Status Event Register: Bits 1-14 = 0 (no event) or 1 (event) for traces 1-14 on the active channel

Command	STATus:QUEStionable:RLIMit:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Ripple Limit Status Condition Register: Bits 1-14 = 0 (pass) or 1 (fail) for traces 1-14 on the active channel

Command	STATus:QUEStionable:RLIMit:ELIMit:CONDition?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Ripple Limit Extra Status Condition Register: Bits 1-2 = 0 (pass) or 1 (fail) for traces 15-16 on the active channel

Command	STATus:QUEStionable:RLIMit:ELIMit:ENABle
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the Questionable Ripple Limit Extra Status Enable Register (for traces 15-16)

Command	STATus:QUEStionable:RLIMit:ELIMit:NTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the negative transition filter of the Questionable Ripple Limit Extra Status Register

Command	STATus:QUEStionable:RLIMit:ELIMit:PTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the positive transition filter of the Questionable Ripple Limit Extra Status Register

Command	STATus:QUEStionable:RLIMit:ELIMit[:EVENt]?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Ripple Limit Extra Status Event Register: Bits 1-2 = 0 (no event) or 1 (event) for traces 15-16 on the active channel

Command	STATus:QUEStionable:RLIMit:ENABle
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the Questionable Ripple Limit Status Enable Register (for traces 1-14)

Command	STATus:QUEStionable:RLIMit:NTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the negative transition filter of the Questionable Ripple Limit Status

Command	STATus:QUEStionable:RLIMit:PTRansition
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the positive transition filter of the Questionable Ripple Limit Status

Command	STATus:QUEStionable:RLIMit[:EVENt]?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Ripple Limit Status Event Register: Bits 1-14 = 0 (no event) or 1 (event) for traces 1-14 on the active channel

6.18.5. STATUS REGISTERS

Enable and query events in the questionable status registers

Command	STATus:QUEStionable:CONDition?
Parameters	int
Scope	Instrument
Summary	Set the Questionable Status Condition Register: Bit 10 = 0 (pass) or 1 (fail) for the overall test

Command	STATus:QUEStionable:ENABle
Parameters	int
Range	0 to 65535
Scope	Instrument
Summary	Set the Questionable Status Enable Register

Command	STATus:QUEStionable:NTRansition
Parameters	int
Summary	Set the negative transition filter of the Questionable Status Register

Command	STATus:QUEStionable:PTRansition
Parameters	int
Summary	Set the positive transition filter of the Questionable Status Register

Command	STATus:QUEStionable[:EVENt]?
Parameters	int
Scope	Instrument
Summary	Read the Questionable Status Event Register: Bit 10 = 0 (overall test event) or 1 (no event)

Command	STATus:PRESet
Scope	Instrument
Summary	Initialize the status registers

7. Programming Examples

7.1. Python

```
import sys
import socket
import time
import struct           # Only needed if retrieving data in real format (rather than ASCII)
#evna_ip = socket.gethostbyname('HOSTNAME')     # Obtain the IP of the host PC by hostname
evna_ip = 'localhost'   # IP address of the host PC (or localhost / 127.0.0.1)
evna_port = 5026        # eVNA Studio requires port 5026

# Function to send SCPI command (no response expected)
def SCPI_Command(scpi):
    s.send(str.encode(scpi + '\n'))                     # Send (add new line and encode)
    print (scpi)

# Function to send SCPI query and get ASCII response
def SCPI_Query(scpi):
    s.send(str.encode(scpi + '\n'))                     # Send (add new line and encode)

    try:
        response_list = []                              # List to collect the response in mutiple chunks
        while True:                                     # Loop to collect each chunk of the response
            response_chunk = s.recv(1024)               # Get the response (up to max buffer size)
            response_list.append(str(response_chunk, 'utf-8'))  # Append it to the list
            if '\n' in str(response_chunk, 'utf-8'):    # Keep looping until new line character received
                break

        evna_response = ''.join(response_list)          # Join the list of responses into a single string

    except Exception as e:
        evna_response = 'Query failed: ' + str(e)  # Check error

    print (scpi, '==>', evna_response)
    return evna_response

# Function to handle SCPI data queries in real 32 format (FORMAT:DATA REAL32)
def SCPI_Query_Real(scpi):
    s.send(str.encode(scpi + '\n'))                     # Send query (add new line and encode)

    try:
        response_list = []                              # List to collect the response in mutiple chunks
        while True:                                     # Loop to collect the response across multiple chunks
            response_chunk = list(s.recv(1024))         # Get a chunk of the response (up to max buffer size)
            response list.extend(response chunk)        # Append the new chunk to the response list
            if 10 in response_chunk:                    # New line char indicates the end of the response
                break

        # Remove a leading formatting byte (#6000080) and the trailing new line char
        trimmed_list = response_list[8:(len(response_list)-1)]

        evna_response = []                              # New list to collect the real data values
        for x in range(0, len(trimmed_list)-1, 4):      # Loop through each group of 4 bytes
            data_bytes = bytes(trimmed_list[x:(x+4)])   # Select the next group of 4 bytes
            data_value = struct.unpack('>f', data_bytes)    # Use struct to unpack the data value
            evna_response.append(data_value[0])         # Add the data value to a list

    except Exception as e:
        evna_response = 'Query failed: ' + str(e)       # Check error

    print (scpi, '==>', evna_response)
    return evna response
```

```
# ---------------------------------------------
# Connect to the eVNA Studio GUI / API on host PC
# ---------------------------------------------

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(3)     # Set timeout in seconds
# Note if using *OPC *OPC? *WAI for synchonisation of operations:
# - Ensure the API socket timeout is set longer than the operation time, to avoid timeout errors

try:
    s.connect((evna_ip, evna_port))         # Connect to eVNA Studio
    print ('Connected to eVNA Studio on host PC:', evna_ip, str(evna_port))
except Exception as e:                      # Connection failed
    print('Could not connect:', evna_ip, evna_port, str(e))
    s.close()
    sys.exit(0)

# ---------------------------------------------
# Find & connect an eVNA-63+ instrument
# ---------------------------------------------

evna_list = SCPI_Query('SYSTEM:DISCOVER?')      # Search for available eVNA instruments, connected to host PC
evna_name = evna_list.split(',')                # Comma separated list of eVNA names
SCPI_Command('SYSTEM:CONNECT ' + evna_name[0])  # Connect the first eVNA found

evna_id = SCPI_Query('*IDN?')                   # Identify the connected eVNA
evna_id_parts = evna_id.split(',')              # Comma separated list of identification parameters
if evna_id_parts[0] != 'EVNA63':                # Confirm a real instrument connected (rather than simulator)
    print("eVNA instrument connection failed")
    s.close()
    sys.exit(0)

SCPI_Command('SYSTEM:PRESET')                   # Reset eVNA configuration to the preset state

# ---------------------------------------------
# Load & save states & data
# ---------------------------------------------

state_filename = "C:/Users\lee.minicircuits\Desktop\evna.cdstate"
snp_filename = "C:/Users\lee.minicircuits\Desktop\evna_snp.s2p"

SCPI_Command('MMEMory:STORe:SALL 1')            # Set state files to save with all channels & traces
SCPI_Query('MMEMory:STORe:SALL?')

SCPI_Command('MMEMory:STORe:STYPe CDState')     # Set state file type as .cdstate (setup + cal + data)
SCPI_Query('MMEMory:STORe:STYPe?')
SCPI_Command(f'MMEMory:STORe:STATe "{state_filename}"')   # Save the state file

SCPI_Command(f'MMEMory:STORe:SNP:DATA "{snp_filename}"')  # Save s2p data

SCPI_Command('SYSTEM:PRESET')                   # Reset eVNA configuration to the preset state
time.sleep(3)

SCPI_Command(f'MMEMory:LOAD:STATe "{state_filename}"');   # Load a pre-existing state file

# ---------------------------------------------
# Channel settings
# ---------------------------------------------

SCPI_Command('SERVice:CHANnel:COUNt 4')         # Enable 4 channels
SCPI_Query('SERVice:CHANnel:COUNt?')            # Check the number of channels

SCPI_Command('SERVice:CHANnel:ACTive 3')        # Set the active channel
SCPI_Query('SERVice:CHANnel:ACTive?')           # Check the active channel

SCPI_Command('DISPlay:SPLit D12_34')            # Arrange the 4 channels on the display
SCPI_Query('DISPlay:SPLit?')                    # Check the display layout
```

```
# ------------------------------------------------
# Trace settings
# ------------------------------------------------

for channel in range(1, 5):                    # Iterate through channels 1 to 4 to set up traces

    SCPI_Command(f'CALCulate{channel}:PARameter:COUNt 4')      # Set number of traces for this channel
    SCPI_Query(f'CALCulate{channel}:PARameter:COUNt?')         # Check number of traces

    #SCPI_Command('DISPlay:CHANnel2:SPLit D12_34')     # Arrange the 4 traces on the display (single channel)
    #SCPI_Query('DISPlay:CHANnel2:SPLit?')             # Check the display layout

    SCPI_Command(f'CALCulate{channel}:PARameter3:SELect')      # Set the active trace
    SCPI_Query(f'SERVice:CHANnel{channel}:TRACe:ACTive?')      # Check which is the active trace

    SCPI_Command(f'CALCulate{channel}:PARameter:DEFine S21')  # Set the measurement type for the active trace
    SCPI_Query(f'CALCulate{channel}:PARameter:DEFine?')        # Check the measurement type for the active
trace

    SCPI_Command(f'CALCulate{channel}:PARameter:SPORt 1')      # Set the source port for the active trace
    SCPI_Command(f'CALCulate{channel}:PARameter:RPORt 2')      # Set the receiver port for the active trace
    SCPI_Query(f'CALCulate{channel}:PARameter:SPORt?')         # Check the source port for the active trace
    SCPI_Query(f'CALCulate{channel}:PARameter:RPORt?')         # Check the receiver port for the active trace

    SCPI_Command(f'CALCulate{channel}:SELected:FORMat PHASe') # Set the data format
    SCPI_Query(f'CALCulate{channel}:SELected:FORMat?')         # Check the data format

    SCPI_Command(f'CALCulate{channel}:SELected:PHASe RADians')    # Set the phase units
    SCPI_Query(f'CALCulate{channel}:SELected:PHASe?')         # Check the phase units

    SCPI_Command(f'DISPlay:CHANnel{channel}:TRACe1:STATe 0') # Enable / disable display of a trace
    SCPI Query(f'DISPlay:CHANnel{channel}:TRACe1:STATe?')     # Check whether a trace is displayed

# ------------------------------------------------
# Marker settings
# ------------------------------------------------

SCPI Command('SYSTEM:PRESET')                                  # Reset eVNA configuration to the preset
state
SCPI_Command('SERVice:CHANnel:COUNt 2')                        # Enable 2 channels
SCPI_Command('SERVice:CHANnel:ACTive 2')                       # Set the active channel
SCPI_Command('CALCulate2:PARameter:COUNt 2')                   # Set 2 traces

SCPI Command('CALCulate2:PARameter2:SELect')                   # Select trace 2 as active

for marker in range(1, 5):

    # Sets 4 markers at 1 to 4 GHz (be sure to select the trace first)

    SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:ACTivate')     # Set the specified marker as the active
marker

    SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:STATe 1')      # Enable the marker display
    SCPI_Query(f'CALCulate2:SELected:MARKer{marker}:STATe?')         # Is the marker displayed?

    SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:COUPle 0')     # Uncouple the marker from other traces
    SCPI_Query(f'CALCulate2:SELected:MARKer{marker}:COUPle?')        # Is the marker coupled across all
traces?

    SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:DISCrete 1')   # Discrete mode (marker fixed to sweep
points)
    SCPI_Query(f'CALCulate2:SELected:MARKer{marker}:DISCrete?')      # Is the marker in discrete mode?

    SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:X {marker}e9')  # Set the marker frequency (GHz)

    SCPI_Query(f'CALCulate2:SELected:MARKer{marker}:X?')             # Query the marker frequency
    SCPI_Query(f'CALCulate2:SELected:MARKer{marker}:Y?')             # Query the marker value

    SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:REFerence:STATe 0')    # Disable reference / delta mode
(values relative to marker 10)
    SCPI_Query(f'CALCulate2:SELected:MARKer{marker}:REFerence:STATe?')       # Check reference mode

    #SCPI_Command(f'CALCulate2:SELected:MARKer{marker}:SET CENTer')
    time.sleep(3)
```

```
# ------------------------------------------------
# Marker limit testing
# ------------------------------------------------

test_pass_fail = SCPI_Query('CALCulate:LIMit:FAIL?')    # Check the limit test pass / fail status

# ------------------------------------------------
# Trigger settings
# ------------------------------------------------

SCPI_Command('ABORt')                           # Abort the current sweep
SCPI_Command('INITiate:CONTinuous 1')           # Set continuous mode
SCPI_Query('INITiate:CONTinuous?')              # Check continuous / hold mode
#SCPI_Command('INITiate:IMMEDiate')              # Initiate new trigger (in certain conditions)

SCPI_Command('TRIGger:SEQuence:SOURce BUS')     # Enable SCPI trigger source
SCPI_Query('TRIGger:SEQuence:SOURce?')          # Check the trigger source
SCPI_Command('TRIGger:SEQuence:SINGle')         # Send SCPI trigger
#SCPI_Command('TRIGger:SEQuence:IMMediate')      # Trigger immediately (in manual or SCPI mode)

SCPI_Command('TRIGger:SEQuence:POINt 0')        # Disable point trigger
SCPI_Query('TRIGger:SEQuence:POINt?')           # Check point / sweep trigger

SCPI_Command('TRIGger:SEQuence:SCOPe ACTive')   # Set the trigger to the active channel only
SCPI_Query('TRIGger:SEQuence:SCOPe?')           # Check the trigger scope

SCPI_Command('TRIGger:EXTernal:DELay 5e-9')     # Set an external trigger delay
SCPI_Query('TRIGger:EXTernal:DELay?')           # Check the external trigger delay

SCPI_Command('SENSe:AVERage:COUNt 8')           # Set the average count (number of sweeps to average per
trigger)
SCPI Query('SENSe:AVERage:COUNt?')              # Check the average count
SCPI_Command('TRIGger:AVERage 1')               # Enable trigger averaging
SCPI_Query('TRIGger:AVERage?')                  # Check trigger averaging

# ------------------------------------------------
# Frequency sweep settings
# ------------------------------------------------

SCPI_Command('SENSe:SWEep:POINts 10')           # Set number of points in the sweep
SCPI_Query('SENSe:SWEep:POINts?')

SCPI_Command('SENSe:FREQuency:START 1000000000')    # Set start frequency
SCPI Command('SENSe:FREQuency:STOP 2000000000')     # Set stop frequency

SCPI_Query('SENSe:FREQuency:START?')
SCPI_Query('SENSe:FREQuency:STOP?')

#SCPI_Query('SENSe:FREQuency:CENTer?')
#SCPI_Query('SENSe:FREQuency:SPAN?')

SCPI_Query('SENSe:FREQuency:DATA?')             # Read list of discrete frequency points set for the sweep
```

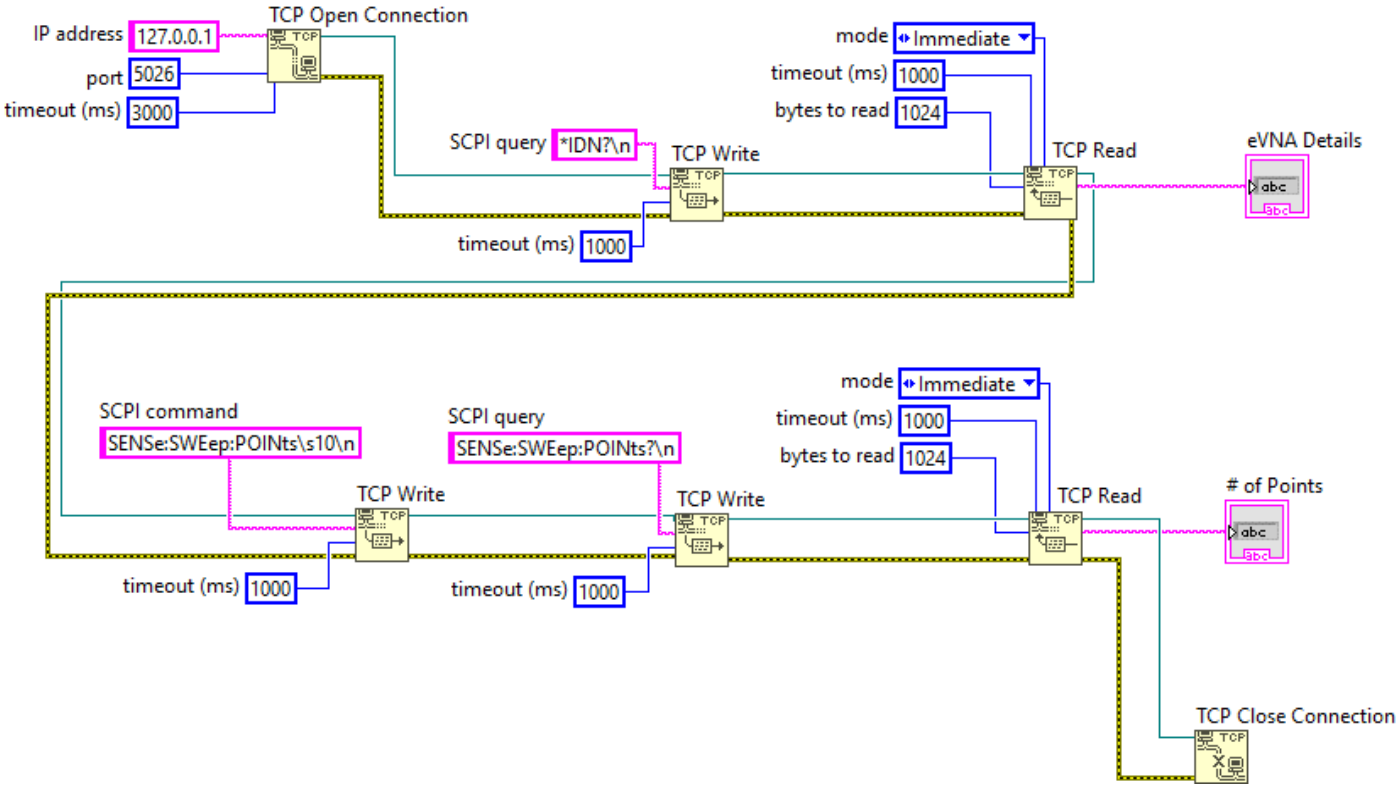
```
# ------------------------------------------------
# Retrieve measurement data arrays
# ------------------------------------------------

SCPI_Command('FORMAT:DATA ASC')                 # Set ASCII data format
SCPI_Query('FORMAT:DATA?')

SCPI_Query('CALCulate:SELected:DATA:RDATa?')    # Retrieve the raw measurement data
SCPI_Query('CALCulate:SELected:DATA:SDATa?')    # Retrieve the corrected measurement data
SCPI_Query('CALCulate:SELected:DATA:FDATa?')    # Retrieve the formatted measurement data

# Set the raw / corrected / formatted measurement data (note: supply 2 data points per sweep point)
#SCPI_Command('CALCulate:SELected:DATA:RDATa 0.000755372,0.000668244,0.000166974,0.000329551,-2.30479e-05,-
0.00152987,-0.000350634,-0.000882104,-0.00142575,-0.000169225,0.00115221,-0.0015869,-
0.000880961,0.000218272')
#SCPI_Command('CALCulate:SELected:DATA:SDATa 0.000752236,0.000668967,0.000166862,0.00032973,-2.2865e-05,-
0.00152912,-0.00035028,-0.000882487,-0.00142602,-0.000168519,0.00115259,-0.00158577,-
0.000880764,0.000217723')
#SCPI_Command('CALCulate:SELected:DATA:FDATa -59.9229,0,-68.6565,0,-56.3063,0,-60.4493,0,-56.8599,0,-
54.1527,0,-60.8429,0')

SCPI_Query('CALCulate:SELected:DATA:SMEMory?')    # Retrieve the corrected memory data
SCPI_Query('CALCulate:SELected:DATA:FMEMory?')    # Retrieve the formatted memory data

# Set the corrected / formatted memory data (note: supply 2 data points per sweep point)
#SCPI_Command('CALCulate:SELected:DATA:SMEMory -0.997196,0.0248926,0.330012,0.928627,0.896212,-
0.405624,0.396256,-0.910373,-0.667643,-0.734523,-0.971169,0.161981,-0.433592,0.876005')
#SCPI_Command('CALCulate:SELected:DATA:FMEMory -10,0,-10,0,-10,0,-10,0,-10,0,-10,0,-10,0,-10,0')

SCPI_Command('FORMAT:DATA REAL32')
SCPI_Query('FORMAT:DATA?')
# Refer to SCPI Query Real function definition for a method to unpack the response
SCPI_Query_Real('CALCulate:SELected:DATA:FDATa?')    # Retrieve the formatted measurement data

# ------------------------------------------------
# End the program
# ------------------------------------------------

SCPI_Command('SYSTEM:DISCONNECT')               # Disconnect the active eVNA instrument
s.close()
print ('Disconnected')
```

7.2. LabVIEW

The example below illustrates the basic steps to connect to the eVNA using the LabVIEW TCP Read / Write functions. The process below can be expanded to send any of the SCPI commands / queries detailed above.

The key considerations are:

1. IP address 127.0.0.1 and port 5026 used if the VI is on the same host PC as the eVNA Studio software
2. String constants are in `'\' Codes Display` format, using `\n` to terminate the command / query
3. `TCP Read` mode is set to `Immediate`
4. SCPI commands can be sent using just a `TCP Write` as there is no response to check
5. SCPI queries use `TCP Write` to send the query, followed by `TCP Read` to collect the response



8. SCPI Error Codes

Summary of error codes which may be returned by the eVNA

8.1. IEEE Common Error Codes

Code	Message	Description
-100	Command syntax error	SCPI command error
-101	Invalid character	Command contains an invalid character
-102	Syntax error	General SCPI syntax error
-103	Invalid separator	Delimiter expected, different character sent
-104	Data type error	A wrong data element type was received
-105	Parameter not allowed	Number of parameters exceeds command syntax
-106	Missing parameter	Number of parameters is less than command syntax
-107	Command not supported	The command has a valid SCPI syntax but not supported by the instrument
-108	Header suffix out of range	The unit of the header is out of range
-110	Numeric data error	Numeric parameter has invalid syntax
-111	Invalid character in number	A character is invalid in a numeric parameter - e.g. letter found in a decimal value
-112	Exponent too large	Absolute value of the exponent exceeds 32,000
-113	Too many digits	More than 255 digits in a number
-114	Numeric data not allowed	A numeric data element was received at an invalid position
-120	Invalid suffix	Suffix has invalid syntax or value
-121	Suffix too long	Unit has more than 11 characters
-122	Suffix not allowed	A suffix is attached to a numeric value which does not have units
-130	Invalid character data	An invalid character found in a character data element
-131	Character data not allowed	A character data element was received at an invalid position
-140	String data error	The content of the string has syntax error
-141	Invalid string data	The character string is invalid
-142	String data not allowed	A character string data element was received at an invalid position
-143	String too long	String longer than 255 characters
-150	Invalid block data	e.g. block data length mismatch
-151	Block data not allowed	A block data element was received at an invalid position
-160	Expression error	Expression syntax error
-161	Invalid expression	The expression data element is invalid
-162	Expression data not allowed	An expression data element was received at an invalid position
-200	Execution error	Error in execution
-201	Measurement in progress - init ignored	Init command received while measurement is in progress. Init command is ignored
-210	Parameter error	General parameter error
-211	Unexpected trigger - ignored	A trigger is detected when not in "waiting for trigger" state
-212	Data out of range	Data element out of range was received

-214	Illegal parameter value	Measurement parameter is invalid - e.g. the source or receive port are invalid
-220	File not found	Specified file not found
-221	File name error	File name syntax error
-300	System error	General system error
-301	Error queue overflow	An error occurred when there is only one slot available in the error queue
-400	Query error	General query error
-401	Querry interrupted	Data bytes are received before a response of a previous query
-402	Query unterminated	Instrument is designated as the talker and an incomplete program message is received
-403	Query deadlocked	Both input and output buffers are full
-404	Query unterminated after indefinite response	Query asking for indefinite response is followed by a new query

8.2. Instrument Specific Error Codes

Code	Message	Description
10	Cannot calculate calibration data - missing standards	Not all the standards needed for the selected calibration method were measured
11	Duplicate port numbers	Identical port numbers specified in a list of ports
12	Calibration method not selected	Trying to save calibration data before method is selected
13	Specified error term is invalid for the selected calibration method	Trying to read an error term which is N/A for the selected calibration method
14	Normal calibration not allowed in frequency offset mode	Tried to execute normal calibration in frequency offset mode
15	Scalar mixer calibration not allowed if frequency offset mode disabled	Tried to execute scalar mixer calibration when frequency offset mode is disabled
16	Partial override with invalid or unspecified calibration method	Tried to perform partial override when the calibration method is not specified
17	Correction not enabled - no calibration data exists	Error correction can't be enabled - no calibration data exists
20	eCal module does not support specified number of ports	Tried to run multi-port calibration where the connected eCal module has less ports
21	eCal failure	Failure to configure or read from eCal module
22	eCal not connected in RF path	eCal auto-detect does not recognize RF connectivity between the VNA and the eCal module
23	Mixed mode S parameters cannot be confidence checked	Tried to perform confidence check for mixed mode S parameters
24	Characterization not found in eCal module	Tried to read a non-existing characterization entry from eCal module memory
30	Target value not found	Target value could not be found (within the specified excursion) in target search
31	Peak not found	Peak not found (within the specified excursion) in peak search
40	Specified a channel which is not on display	Tried to select an active channel which is not displayed
41	Specified trace does not exist	Tried to select an active trace which does not exist
42	No valid memory trace	Tried to perform memory operation where there is no valid memory trace
43	Time domain processing not supported in frequency offset mode	Tried to perform time domain transform or gating in frequency offset mode
44	Fixture Simulator not allowed in frequency offset mode	Tried to run fixture simulator in frequency offset mode

45	Auto port extension not allowed in power sweep or frequency offset mode	Tried to enable auto port extension in power sweep or frequency offset mode
46	No valid measurement to save in file	Tried to save trace data where no valid trace is available
50	Frequency out of range	Tried to select a frequency outside the frequency range of the instrument
60	Power meter not found	Failed to locate power meter based on specified VISA resource name
61	Power meter not stable	Power meter reading does not stabilize within pre-defined time, or reading exceeds tolerance
62	Signal generator not found	Failed to located signal generator based on specified VISA resource name
63	Signal generator control failure	Failure to control external signal generator
100	Load failed	Load from file failed
101	Save failed	Save to file failed
102	THRU standard should be defined with S2P file	THRU standard should be defined with S2P file
103	Standard should be defined with S1P file	Standard should be defined with S1P file
200	Option not installed	Tried to perform an optional function when the required option is not installed in the instrument
210	PLL not locked	At least one frequency point resulted in unlocked PLL. Measurement data is not valid
220	Power trip event occurred - turned RF off	Measurement detected too high receive power. Source is turned off to prevent damage
230	Self test failure	Instrument self test failed
300	Adapter length cannot be estimated in zero span	Tried to perform adapter length estimation in zero span
400	Invalid equation expression	Invalid equation is specified in the equation editor
401	Invalid equation label	Invalid equation label e.g. contains space
500	Instrument disconnected	
501	Instrument is not connected	Tried to connect to instrument that was not connected to host
502	Instrument connection failed	Instrument connection failed
503	Instrument connection failed	Instrument already claimed
504	Instrument connection failed	Instrument not activated
505	Instrument connection failed	Device is not connected to USB
506	USB Storage set state failure	Failed to set state of USB Storage
507	USB Storage get state failure	Failed to get state of USB Storage

508	Instrument activation failed	Instrument already claimed
509	Instrument activation failed	Instrument already activated
510	Instrument activation failed	Device is not connected to USB
700	Insufficient memory for current sweep settings	Insufficient memory for current sweep settings. Sweep recording wasn't started.

8.3. Warnings

Code	Message	Description
1000	Calibration interpolated	Measurement and calibration stimulus points are not identical. Calibration data was interpolated
1001	Measurement conditions not identical to calibration	Measurement and calibration conditions (e.g. power, IF BW) are not identical. Error correction might not be accurate
1002	Changed global power or IF BW - updated segment table accordingly	Global power or IF BW changed - the per-segment power / IF BW were overridden in the segment table

8.4. Notifications

Code	Message	Description
2000	Instrument connected	

9. Contact

Mini-Circuits

13 Neptune Avenue

Brooklyn, NY 11235

Phone: +1-718-934-4500

Email: sales@minicircuits.com

Web: www.minicircuits.com

Important Notice

Trademarks