Programming Manual

ShockLine™ MS46121A/B, MS46122A/B, MS46322A/B Series Vector Network Analyzers

MS46121A/B-004 VNA, 40 MHz to 4 GHz, 1-Port
MS46121A/B-006 VNA, 150 kHz to 6 GHz, 1-Port

MS46122A/B-010 VNA, 1 MHz to 8 GHz, 2-Port
MS46122A/B-020 VNA, 1 MHz to 20 GHz, 2-Port
MS46122A/B-040 VNA, 1 MHz to 43.5 GHz, 2-Port

MS46322A-004 VNA, 1 MHz to 4 GHz, 2-Port
MS46322A/B-010 VNA, 1 MHz to 8 GHz, 2-Port
MS46322A-014 VNA, 1 MHz to 14 GHz, 2-Port
MS46322A/B-020 VNA, 1 MHz to 20 GHz, 2-Port
MS46322A-030 VNA, 1 MHz to 30 GHz, 2-Port
MS46322A/B-040 VNA, 1 MHz to 43.5 GHz, 2-Port

Anritsu

Anritsu Company
490 Jarvis Drive
Morgan Hill, CA 95037-2809
USA

Part Number: 10410-00338
Revision: U
Published: March 2017
Copyright 2017 Anritsu Company

Table of Contents

Chapter 1 — General Information

1-1 About This Manual 1-1
Remote Programming 1-1
1-2 Introduction 1-2
1-3 IEEE 488 Description 1-2
1-4 Third Party Programming and Driver Support 1-2
1-5 Ethernet LAN TCP/IP Description 1-3
TCP/IP General Requirements and Settings 1-3
Default Plug-and-Play Configuration 1-4
Manually Configuring TCP/IP Ethernet LAN Settings 1-4
Connecting the MS46122A/B, MS46121A/B Remotely using Python 3.4 SOCKETS 1-6
Creating a Script Using an MS46122A/B with Python 3.4 SOCKETS 1-7
Connecting the MS46322A/B to Python 3.4 with a controller PC and Ethernet interface . . 1-9
Creating a Script using an MS46322A/B using Python 3.4 and PyVISA 1-11
Document Conventions 1-12
1-6 Minimum/Maximum Instrument Frequency and Related Parameters 1-13
Model MS46121A/B, MS46122A/B, MS46322A/B Standalone VNA Frequency Limits . . 1-13
Standalone VNAs – Default Start, Default CW, and Default Stop Frequencies 1-13
Standalone VNAs – Minimum Start, Minimum CW, and Maximum Start Frequencies . . . 1-13
Standalone VNAs – Default Frequency Span and Maximum Frequency Span 1-14
Standalone VNAs – Minimum Center and Maximum Center Frequencies 1-15
Standalone VNAs – Default Center Frequencies 1-15

Chapter 2 — Programming the ShockLine™ Series VNA

2-1 Introduction 2-1
2-2 Introduction to SCPI Programming 2-1
Command Types 2-1
2-3 IEEE 488.2 Commands 2-2
2-4 System Commands 2-2
2-5 SCPI Commands 2-2
Required SCPI Commands 2-2
Native SCPI Commands 2-3
2-6 Command Requirements 2-3
Query Commands 2-3
Command Names 2-3
Hierarchical Command Structure 2-4
Data Parameters 2-4
2-7 Notational Conventions 2-5
General Notations 2-5
Parameter Notations 2-6
Notational Examples 2-7
2-8 Numeric Data Suffix Reference 2-8

Table of Contents (Continued)

2-9 Data Transmission Methods 2-9
<NR1> 2-9
<NR2> 2-9
<NR3> 2-9
<NRf> 2-9
<string> 2-10
<ASCII> or <Arbitrary ASCII> 2-10
<block> or <arbitrary block> 2-10
<char> 2-12
MPND 2-12
MPNF. 2-12
MPNI 2-12
Formatting Data Output 2-12
ASCII or Binary Data Format 2-13
2-10 Calculating the Byte Size 2-15
Numbers Output-per-Data Point (NODP) 2-15
Bytes Output-per-Number (BOPN) 2-15
Size of Data Block (SODB) 2-16
Number of Bytes Output (NBO) 2-16
2-11 Input Buffer Size and NRFD Holdoff 2-16
2-12 Synchronization of Commands 2-17
2-13 Forcing the Parser to Stop Waiting 2-17
2-14 Aborting an RF or Hardware Calibration 2-17
2-15 Time-Out Settings 2-17
2-16 Trace Type Parameters and Coefficients 2-19
2-17 Input/Output Data Files 2-22
2-18 Status System Reporting 2-24
Status Group Registers 2-24
Status Group Reporting 2-24
2-19 Trigger System 2-30
Trigger Modes 2-30
2-20 Calibration Component Parameters 2-31
Loads and Through Lines 2-31
Other Connector Coefficients 2-31
2-21 Calibration Command Overview 2-32
Setting Up a Two-Port Calibration 2-32
Defining the Calibration Standards 2-33
Performing the Calibration 2-36
AutoCal 2-38

2-22 Command Script Example – Limit Lines 2-40
Limit Lines for Single Rectilinear Trace Display 2-40
Required Equipment 2-41
Prerequisites 2-41
DUT Requirements 2-41
Channel and Trace Display Requirements 2-41
VNA General Setup and Configuration 2-42
Frequency and Sweep Settings 2-43
Limit Lines Setup 2-43
Clear Previous Limit Lines 2-44
Create and Configure Limit Line Segment 1 2-44
Create and Configure Limit Line Segment 2 2-44
Create and Configure Limit Line Segment 3 2-45
Create and Configure Limit Line Segment 4 2-45
Configure AutoCal Calibration 2-46
Ready for Measurements 2-46

Chapter 3 — IEEE Commands

3-1 Introduction 3-1
3-2 Command Descriptions 3-1
3-3 Numeric Limits 3-2
3-4 IEEE 488.2 Commands 3-2

Chapter 4 — Diagnostic and Troubleshooting Commands

4-1 Introduction 4-1
4-2 Parameters and Notations 4-1
4-3 Numeric Limits 4-2
4-4 Self Test Commands 4-2

Chapter 5 — SCPI Commands: 1-Port and 2-Port VNAs

5-1 Introduction 5-1
5-2 MS46322A/B vs. MS46122A/B, MS46121A/B SCPI Configuration 5-1
5-3 Minimum/Maximum Frequency Limits and Related Parameters 5-1
5-4 Command Level Hierarchy 5-2
Command Descriptions and Notation Conventions 5-2
Numeric Limits 5-2
Notational Conventions 5-2
5-5 General Parameters 5-3
5-6 :CALCulate{1-16}:CORRection Subsystem 5-5
Calibration Option Subsystems 5-5
5-7 :CALCulate{1-16}:EXTRaction Subsystem 5-7
Calibration Option Subsystems 5-7
General Parameters 5-7
5-8 :CALCulate{1-16}:DISPlay:MARKer Subsystem 5-14
Marker Subsystems 5-14

Table of Contents (Continued)

5-9 :CALCulate{1-16}:EOOE: Subsystem 5-15

5-10 :CALCulate{1-16}:FORMat Subsystem - SnP Data 5-29
I/O Configuration and File Operation Subsystems 5-29

5-11 :CALCulate{1-16}:FSIMulator:NETWork Subsystem 5-30
Calibration Simulation Subsystems 5-30

5-12 :CALCulate{1-16}:FSIMulator:NETWork {1-50} Subsystem 5-36
Calibration Simulation Subsystems 5-36

5-13 :CALCulate{1-16}:IMPedance:TRANsformation Subsystem 5-41
Calibration Setup Subsystems 5-41

5-14 :CALCulate{1-16}:MARKer Subsystem 5-43
Marker Subsystems 5-43

5-15 :CALCulate{1-16}:PARameter and :PARameter{1-16} 5-44
Trace Subsystems 5-44

5-16 :CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem 5-50
Marker Subsystems 5-50

5-17 :CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem 5-53
Marker Subsystems 5-53

5-18 :CALCulate{1-16}:PARameter{1-16}:SELect Subsystem 5-55
Trace Subsystems 5-55

5-19 :CALCulate{1-16}:POLar Subsystem 5-56
Trace Subsystems 5-56

5-20 :CALCulate{1-16}:PROCessing:ORDer Subsystem 5-58
Time Domain, Group Delay, and Reference Plane Subsystems 5-58

5-21 :CALCulate{1-16}:REFerence Subsystem 5-59
Calibration Setup Subsystems 5-59
Time Domain, Group Delay, and Reference Plane Subsystems 5-59

5-22 :CALCulate{1-16}[:SELected]:CONVersion Subsystem 5-63
Trace Subsystems 5-63

5-23 :CALCulate{1-16}[:SELected]:DATA Subsystem 5-65
I/O Configuration and File Operation Subsystems 5-65
Trace Subsystems 5-65

5-24 :CALCulate{1-16}[:SELected]:FORmat Subsystem 5-67
Trace Subsystems 5-67

5-25 :CALCulate{1-16}[:SELected]:LIMit Subsystem 5-69
Limit Line and Segment Subsystems 5-69
Trace Subsystems 5-69

5-26 :CALCulate{1-16}[:SELected]:MARKer Subsystem 5-73
Marker Subsystems 5-73

5-27 :CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem 5-88
Marker Subsystems 5-88
Trace Subsystems 5-88

5-28 :CALCulate{1-16}[:SELected]:MATH Subsystem 5-91
Trace Subsystems 5-91

5-29 :CALCulate{1-16}[:SELected]:MDATA Subsystem 5-94
Trace Subsystems 5-94
5-30 :CALCulate{1-16}[:SELected]:SMOothing Subsystem 5-95
Trace Subsystems 5-95
5-31 :CALCulate{1-16}[:SELected]:TDATA Subsystem 5-96
Trace Subsystems 5-96
I/O Configuration and File Operation Subsystems 5-96
5-32 :CALCulate{1-16}[:SELected]:TRANsform:TIME Subsystem 5-98
Time Domain, Group Delay, and Reference Plane Subsystems 5-98
Trace Subsystems 5-98
5-33 :DISPlay Subsystem 5-107
Trace Subsystems 5-107
Marker Subsystems 5-107
Limit Line Subsystems 5-107
5-34 :FORMat Subsystem 5-122
I/O Configuration and File Operation Subsystems 5-122
5-35 :HCOPy Subsystem 5-125
I/O Configuration and File Operation Subsystems 5-125
5-36 :MMEMory Subsystem 5-130
I/O Configuration and File Operation Subsystems 5-130
5-37 :SENSe{1-16}:ABORtcal Subsystem 5-136
Calibration Subsystems with Actual Calibrations 5-136
5-38 :SENSe{1-16}:AVERage Subsystem 5-136
Sweep Subsystems 5-136
5-39 :SENSe{1-16}:BANDwidth Subsystem 5-138
IF Configuration Subsystems 5-138
5-40 :SENSe{1-16}:BWIDth Subsystem 5-139
IF Configuration Subsystems 5-139
5-41 :SENSe{1-16}:CORRection:COEFficient:PORT 5-140
Calibration Simulation Subsystems 5-140
Calibration Type Abbreviations 5-140
Related Query 5-140
5-42 :SENSe{1-16}:CORRection:COEFficient Subsystem 5-144
Calibration Simulation Subsystems 5-144
Calibration Type Abbreviations 5-144
5-43 :SENSe{1-16}:CORRection:COLLect:ECAL Subsystems 5-148
5-44 :SENSe{1-16}:CORRection:COLLect:METHod Subsystem 5-151
Calibration Setup Subsystems 5-151
5-45 :SENSe{1-16}:CORRection:COLLect:MICrostrip Subsystem 5-152
Calibration Setup Subsystems 5-152
5-46 :SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem 5-156
Calibration Setup Subsystems 5-156

5-47 :SENSe{1-16}:CORRection:COLLect:PORT Subsystem 5-157
Calibration Subsystems with Actual Calibrations. 5-157
Calibration Type Abbreviations 5-157
5-48 :SENSe{1-16}:CORRection:COLLect Subsystem 5-188
Calibration Setup Subsystems. 5-188
5-49 :SENSe{1-16}:CORRection:COLLect[:CALa]:PORT 5-189
Calibration Subsystems with Actual Calibration 5-189
Calibration Abbreviations. 5-189
5-50 :SENSe{1-16}:CORRection:COLLect Subsystem 5-191
Calibration Setup Subsystems. 5-191
Related Command Subsystems 5-191
Calibration Method Names 5-191
5-51 :SENSe{1-16}:CORRection:COLLect:WAVeguide 5-199
Calibration Setup Subsystems. 5-199
5-52 :SENSe{1-16}:CORRection:EXTension Subsystem 5-206
Time Domain, Group Delay, and Reference Plane Subsystems. 5-206
5-53 :SENSe{1-16}:CORRection:ISOLation Subsystem. 5-207
Calibration Setup Subsystems. 5-207
5-54 :SENSe{1-16}:CORRection:STATe Subsystem 5-208
Calibration Setup Subsystems. 5-208
5-55 :SENSe{1-16}:FREQuency Subsystem 5-209
Sweep Subsystems 5-209
Frequency Limits 5-209
5-56 :SENSe{1-16}:FSEGMent Subsystem 5-211
Limit Line and Segment Subsystems 5-211
5-57 :SENSe{1-16}:FSEGMent{1-50} Subsystem 5-218
Limit Line and Segment Subsystems 5-218
5-58 :SENSe{1-16}:HOLD Subsystem 5-222
Trigger, Hold, and External Source Subsystems. 5-222
5-59 :SENSe{1-16}:ISEGMent Subsystem 5-225
Limit Line and Segment Subsystems 5-225
5-60 :SENSe{1-16}:ISEGMent{1-50} Subsystem 5-231
Limit Line and Segment Subsystems 5-231
5-61 :SENSe{1-16}:RECEiver Subsystem 5-235
5-62 :SENSe{1-16}:SEGMent Subsystem 5-235
Limit Line and Segment Subsystems 5-235
5-63 :SENSe{1-16}:SWEep Subsystem 5-236
Sweep Subsystems 5-236
5-64 :SOURce{1-16}:POWer Subsystem 5-238
Power Configuration Subsystems 5-238
5-65 :STATus:OPERation Subsystem 5-239
5-66 :STATus:QUEStionable Subsystem 5-241
5-67 :SYSTem Subsystem 5-244

Appendix A — Agilent ENA SCPI Programming Emulation

A-1 Introduction A-1
A-2 ENA Command Listing A-1

Appendix B — IVI Functions

B-1 Introduction B-1
Error and Status Information B-1
IVI Compliance Information [Agilent Source] B-2
IVI-C Driver Information B-3
Functions and Attributes General Information B-3
Functions and Their Corresponding IVI Class B-4
Relevant Files B-7
B-10 ANVNA_ChannelMeasurementSetUserDefinedParameter B-31
B-11 ViStatus ANVNA_ChannelMeasurementGetUserDefinedParameter B-32
B-12 ANVNA_ChannelMeasurementGetResponseType B-34
B-13 ANVNA_ChannelMeasurementGetMixedModeResponseType B-36
B-14 ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse B-37
B-15 ANVNA_ChannelMeasurementGetMixedModeTwoDiffPairsResponse B-39
B-16 ANVNA_ChannelMeasurementSetMixedModeOneDiffPairOneSingResponse B-42
B-17 ANVNA_ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse B-44
B-18 ANVNA_ChannelMeasurementSetMixedModeOneDiffPairTwoSingResponse B-47
B-19 ANVNA_ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse B-50
B-20 ANVNA_ChannelMeasurementSetMixedModeOneDiffPairResponse B-52
B-21 ANVNA_ChannelMeasurementGetMixedModeOneDiffPairResponse B-55
B-22 ANVNA_ChannelMeasurementSetMaxEfficiencyResponse B-57
B-23 ANVNA_ChannelMeasurementGetMaxEfficiencyResponse B-60
B-28 ANVNA_SetupManualCalibration B-72
B-29 ANVNA_SetManualCalKit B-74
B-31 ANVNA_GetManualCalKit B-80
B-32 ANVNA_LoadCalibrationKit B-83
B-95 ANVNA_ChannelMeasurementGetResponseType B-210
B-96 ANVNA_ChannelMeasurementGetResponseType B-211
B-105 ANVNA_GetMarkerUpLowValue B-226
B-132 ANVNA_GetTimeDomainDistanceValues B-259
B-133 ANVNA_GetTimeDomainGateValues B-260
B-151 ANVNA_GetLowerTraceLowerLimitBuffer B-288
B-152 ANVNA_GetLowerTraceUpperLimitBuffer B-289
B-153 ANVNA_GetLowerTraceLowerLimitFailPointsBuffer B-291
B-154 ANVNA_GetLowerTraceUpperLimitFailPointsBuffer B-292
B-155 ANVNA_SetRippleLimitTestingOnOff B-293
B-156 ANVNA_GetRippleLimitTestingOnOff B-295

B-157 ANVNA_SetRippleLimitTestResultSign B-296
B-158 ANVNA_GetRippleLimitTestResultSign B-297
B-159 ANVNA_AddDefaultRippleLimitSegment B-298
B-160 ANVNA_GetRippleLimitsCount. B-299
B-161 ANVNA_DeleteRippleLimitSegment B-300
B-162 ANVNA_ClearAllRippleLimits B-301
B-163 ANVNA_AddRippleLimitSegment B-302
B-164 ANVNA_GetRippleLimitValues B-304
B-165 ANVNA_DeleteRippleLimitSegmentAt B-306
B-166 ANVNA_IsRippleLimitTestPass B-307
B-167 ANVNA_SetRippleLimitX1Val. B-308
B-168 ANVNA_GetRippleLimitX1Val B-309
B-169 ANVNA_SetRippleLimitX2Val. B-310
B-170 ANVNA_GetRippleLimitX2Val B-311
B-171 ANVNA_SetRippleLimitRippleVal B-312
B-172 ANVNA_GetRippleLimitRippleVal. B-313
B-173 ANVNA_GetRippleLimitUpperLowerValues B-314
B-174 ANVNA_GetRippleLimitFailPointsBuffer. B-315
B-175 ANVNA_GetRippleLimitLineActive B-317
B-176 ANVNA_SetRippleLimitLineActive B-318
B-177 ANVNA_SetRippleLimitLinesOnOff B-319
B-178 ANVNA_GetRippleLimitLinesOnOff B-320
B-179 ANVNA_SetRippleLimitRippleValueFormat B-321
B-180 ANVNA_GetRippleLimitRippleValueFormat B-322
B-181 ANVNA_GetRippleLimitRippleMeasurementValue. B-323
B-182 Attributes. B-325
Attribute Information for the Following Functions B-325
Inherent IVI Attributes B-325
B-183 IVI Driver Installation. B-341

Appendix C — Alphabetical SCPI Command List

C-1 Introduction C-1
Sorting C-1
System Suffix. C-1
C-2 Alphabetical Command List C-1

Chapter 1 — General Information

1-1 About This Manual

ShockLine VNAs support remote operation commanded via the TCP/IP or VXI-11 protocols. This manual provides operation and programming information for this activity. This manual applies to the following instruments:

- MS46121A/B
- MS46122A/B
- MS46322A/B

This document identifies which commands of the following types are supported:

- IEEE488
- SCPI
- IVI-C

This document covers:

- Ethernet connection and setup instructions
- A general description of bus data transfer and control functions
- A listing of the IEEE 488 Interface Function Messages recognized by the VNA
- A brief description of the TCP/IP and VXI-11 program interfaces to the VNA
- A complete listing and description of all the Standard Commands for Programmable Instruments (SCPI) commands that can be used to control VNA operation, with examples of command usage
- applicable IEEE 488.2, System and Troubleshooting, and SCPI commands.

This document should be used together with the Operation Manual for the target instrument. The OM or User Guide provides information about equipment set up and manual operation of the software front panel. Some user interface descriptions here use formats and conventions covered in the User Interface Reference Manual for the target instrument.

Remote Programming

This document covers the ShockLine commands for remote programming of the MS46121A/B, MS46122A/B, and MS46322A/B series ShockLine VNAs. Drivers are required for IVI-C and can be found on any of the ShockLine product webpage Library tabs. Certain commands as well as interconnect setups will not be applicable to some ShockLine VNA models. Those differences are noted in the table below.

Table 1-1. Remote Programming

	MS46322A/B	MS46122A/B	MS46121A/B
Controller Computer	Embedded	External	External
Hardware Communication Interface	LAN	USB	USB
Communication Protocol	TCP/IP Sockets or VXI-1 with NI VISA	TCP/IP Sockets	TCP/IP Sockets
Remote Command Types			
SCPI	Yes	Yes	Yes
IVI-C	Yes	Yes	Yes

1-2 Introduction

This chapter provides a general description of the data transfer and control functions. It also contains a listing of the ShockLine™ software application's interface remote programming capability and response to IEEE 488 interface function messages.

The information presented in this chapter is general in nature. For complete and specific information, refer to the following documents, available from the *Institute of Electrical and Electronics Engineers*:

- ANSI/IEEE Standard 488.1-1987 IEEE Standard Digital Interface for Programmable Instrumentation
- ANSI/IEEE Standard 488.2-1987 IEEE Standard Codes, Formats, Protocols, and Common Commands

These documents precisely define the total specification of the mechanical and electrical interface, and of the data transfer and control protocols.

The final section in this chapter, "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13, provides a listing of the VNA instrument minimum and maximum frequency settings and related parameters such as default frequency span.

Note	When operating the ShockLine VNA through remote programming, on-screen user interface controls are disabled in the ShockLine™ software application. To return to local control, press the keyboard Esc key or send the RTL command. For general information about GPIB, refer to Section 1-3 "IEEE 488 Description".

1-3 IEEE 488 Description

IEEE 488 is an international instrumentation interface standard for integrating instruments, computers, printers, plotters, and other measurement devices into systems. IEEE stands for the Institute of Electrical and Electronics Engineers and is currently the world's largest technical professional society. Refer to http://www.ieee.org for more information about the society and its standards.

1-4 Third Party Programming and Driver Support

For more information on program and driver support not listed elsewhere in this manual, contact Anritsu at ShockLineVNA.support@anritsu.com.

1-5 Ethernet LAN TCP/IP Description

ShockLine™ VNAs support gigabit Ethernet. The instrument or an external computer connects directly to a LAN via an RJ-45 Ethernet Port using a standard CAT-5 Ethernet cable. The instrument or an external computer is programmable through this port by means of IEEE488 and SCPI commands.

The general requirements for manual Ethernet LAN configuration are discussed in the sections below.

Note	ShockLine VNAs do not support USB networking. This section is provided for general information about manually configuring an Ethernet connection. Consult your local network administrator for the exact requirements and settings that are required for your network installation.

TCP/IP General Requirements and Settings

Transmission Control Protocol/Internet Protocol (TCP/IP) is a network protocol. In the Windows 7 operating system, TCP/IP is automatically installed and in most cases, installation, configuration, and communication are transparent to the user.

In a TCP/IP network, you must provide IP addresses and other information to clients. Clients may also require a naming service or a method for name resolution. The TCP/IP protocol setup requires the following information:

- **IP Address**

  Every device in a TCP/IP network requires an IP address that consists of four numbers, each between 0 and 255, separated by periods. For example: 128.111.122.42 is a valid IP address.

- **Subnet Mask**

  The subnet mask distinguishes the portion of the IP address that is the network identification (ID) address from the portion that is the station ID address. When the subnet mask 255.255.0.0 is applied to the IP address above, it would identify the network ID address as 128.111 and the station ID address as 122.42. All stations in the same Local Area Network (LAN) should have the same network ID, but different station IDs.

- **Default Gateway**

  A TCP/IP network can have a gateway to communicate beyond the LAN identified by the network ID. A gateway is a computer or electronic device that is connected to two different networks and can move TCP/IP data from one network to the other. A single LAN that is not connected to other LANs requires a default gateway setting of 0.0.0.0. The default gateway setting for the ShockLine MS46322A/B Series VNA is 0.0.0.0. If your network has a gateway, then the default gateway would be set to the appropriate value of your gateway.

- **Hardware Address (MAC Address)**

  An Ethernet address is a unique 48-bit value that identifies a network interface card internal to the VNA to the rest of the network. Every network card has a unique Ethernet address permanently stored into its memory.

- **TCP/IP Port Number**

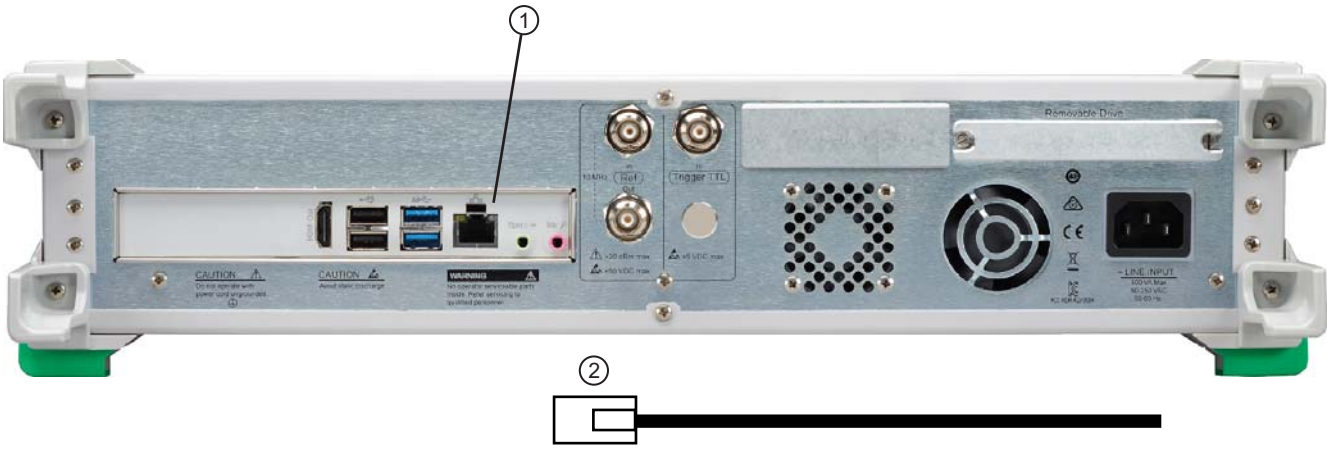
  Reports the currently used TCP/IP port number with Port #5001 set as the default. In general, it should not be changed. If the port number is changed, do not change it to 5000 as that port is used by VXI-11. Ports below #5000 are generally reserved for other services and devices. Custom settings generally should use settings of #5001 and higher.

- **Network Interface Setup**

  TCP/IP connectivity requires setting up the parameters described at the beginning of this section. You may need to contact your network administrator or refer to your network documentation for further assistance. The following procedure is a general overview of how to set up a general LAN connection on both the VNA and the remote machine. The actual menus and sequence may vary.

Default Plug-and-Play Configuration

The ShockLine™ VNA with embedded computer and Windows 7 operating system is pre-configured for connection to any Ethernet network with a gateway and DNS/DHCP. For physical connection, attach as shown below an Ethernet cable between the VNA rear panel RJ-45 Ethernet Port and your local network port. Windows 7 in the instrument automatically detects the network settings and configures the network connection.



1. Ethernet Port – Rear panel Ethernet RJ45 (f) port	2. Ethernet CAT-5 RJ-45 cable from LAN (Local Area Network).

Figure 1-1. Network Connection to MS46322A/B

Manually Configuring TCP/IP Ethernet LAN Settings

To see the current network settings for your VNA, run the ShockLine Application Software and navigate to the Network Interface (Network Interf.) menu as follows:

- MAIN | System | SYSTEM | Network Interface | NETWORK INTERFACE

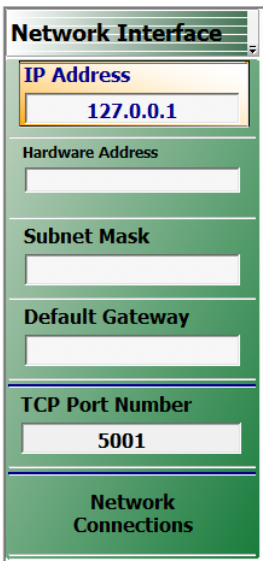


Figure 1-2. NETWORK INTERFACE Menu

The top display buttons provide information for the current network settings. Changes to these settings must be made through the Microsoft Windows 7 configuration utilities reached by clicking the lowest button, Network Connections. This opens the relevant Windows Control Panel item.

The Network Connections dialog box shows the current available local networks and provides access to various network configuration utilities. If connected to one or more networks, a link to each network name is provided with links to the settings of each connection.

Note	You may need to consult your network documentation or network administrator for assistance in manually configuring your network setup. The Microsoft Windows **Network Connections Help** system provides information related to computer networking. If an Internet connection is present, links to Microsoft and other URLs are also provided.

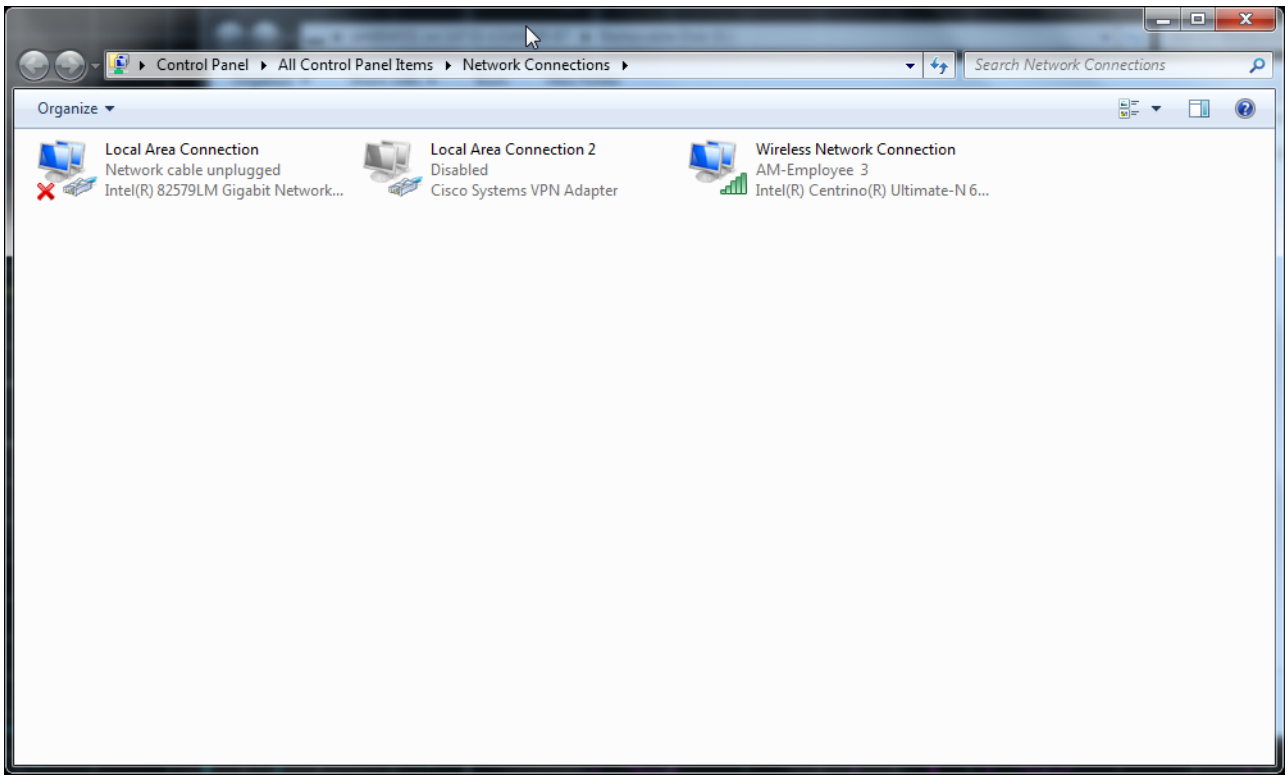


Figure 1-3. Windows NETWORK CONNECTIONS Dialog Box

Connecting the MS46122A/B, MS46121A/B Remotely using Python 3.4 SOCKETS

The MS46122A/B and MS46121A/B both use USB interface to communicate with the PC controller.

Note	When the PC is connected to the USB VNAs, the PC becomes the controller. This means that the PC that is controlling the VNA DOES NOT require another external PC to control it and the VNA. The below example will work on the MS46322A/B in the instance that the Python program is running on the desktop of the MS46322A/B's embedded computer.

1. Download Python 3.4 or the latest edition of Python 3.4 from the Python.org website: https://www.python.org/downloads/
2. In the section for downloads, choose the latest Python edition. i.e. Python 3.4.X
3. Run software and install for all users
4. Open the Windows icon on the PC's desktop and select All Programs
5. Find the folder labeled, "Python 3.4" and open this folder. When this file is open, choose IDLE (Python 3.4 GUI)
6. The Python 3.4.X shell will open and take line by line commands. See Figure 1-4
7. Open the ShockLine application

Note	Note If application is not launched before running Python, the following error will occur: **No connection could be made because the target machine actively refused it.**

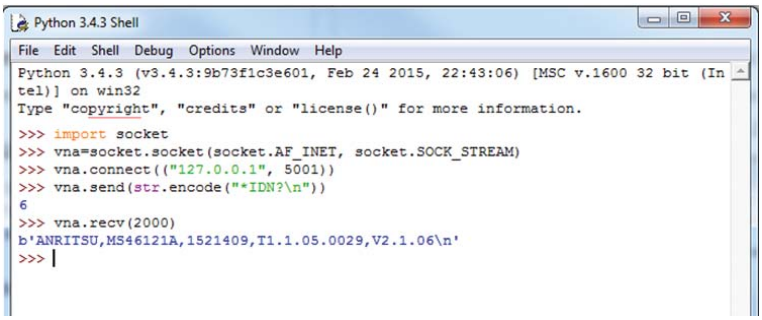


Figure 1-4. Python 3.4.X Shell

8. Type the following commands (CAPS matter). Press "Enter" after each command import socket

- vna=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
- vna.connect(("127.0.0.1", 5001))
- vna.send(str.encode("*IDN?\n"))
- vna.recv(2000)

9. The ShockLine software front panel will now be grayed (locked) out and the response of the *IDN query will be instrument name, serial number, software and firmware version. See Figure 1-5.

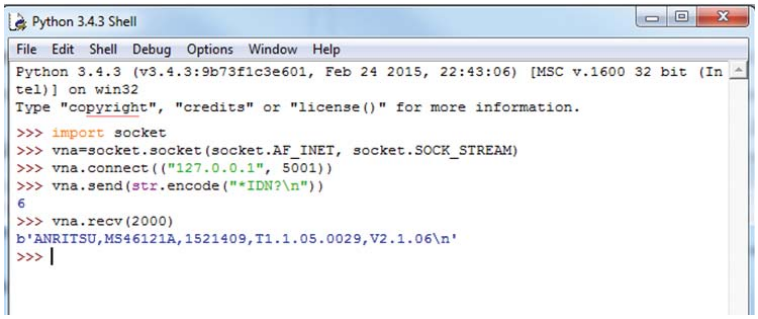


Figure 1-5. Python Shell With IDN? Query

The commands described below:

Import socket-allows the module for socket API to exist

vna=socket.socket(socket.AF_INET, socket.SOCK_STREAM)

VNA is the variable and can be changed to the user's specification. Socket is the module and dot socket is the object instantiation from the socket module. The two arguments open the API for communication.

vna.connect(("127.0.0.1", 5001))

VNA that defines the socket instance and the connect is a function call from the socket module. The address 127.0.0.1 is a local TCP/IP resource loop between the PC and the VNA, and 5001 is the port number that the application is listening on.

vna.send(str.encode("*IDN?\n"))

The send function has a nested str.encode function that converts strings to a specific character set (here it's UTF-8 unless otherwise defined). This is not needed in Python 2.7. The *IDN? String is an instrument query.

vna.recv(2056)

The recv function has an argument that defines the number of bytes Python will receive from the socket until it hits a newline ("\n").

Note	All commands sent to the VNA with SCPI commands will look like what is listed in step 3.d above. The quotation marks and the \n line termination are also required when using python 3.4 sockets. The standard string for SCPI will now look like the string below: vna.send(str.encode**("SCPI Command String\n"))** When using SOCKETS, please make use of the header ("#9_ _ _ _") when querying information as the exact number of bytes will be displayed in the response.

Creating a Script Using an MS46122A/B with Python 3.4 SOCKETS

This script will query the instruments general information along with the start frequency and stop frequencies. It will then set the number of points to 401 and the IFBW to 10 kHz and set trace 4 to Smith Chart.

1. Open the ShockLine application.

2. Open Python 3.4.3 Shell. This is labeled Python IDLE in the Python 3.4 folder.

3. In the Python Shell, choose File option near the header and select New File.

4. An Untitled file will open and it will be blank.
5. Type in the following commands:

```
import socket
vna = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
vna.connect(("127.0.0.1", 5001))
vna.send(str.encode("*IDN?\n"))
reply=vna.recv(2056)
print (reply)
vna.send(str.encode("SENS:FREQ:STAR?\n"))
reply=vna.recv(2056)
print (reply)
vna.send(str.encode("SENS:FREQ:STOP?\n"))
reply=vna.recv(2056)
print (reply)
vna.send(str.encode("SENS:SWEEP:POINT 401\n"))
vna.send(str.encode("SENS:BAND 70E1\n"))
vna.send(str.encode("CALCulate1:PARameter4:FORMat SMITh\n"))
```

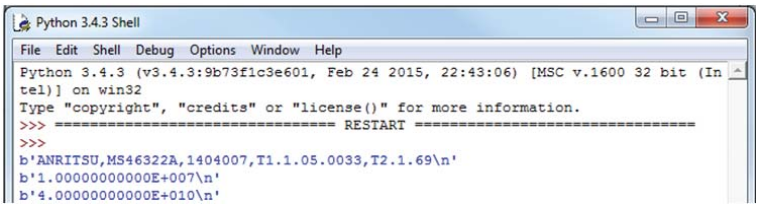


Figure 1-6. Python Script with SCPI Commands for the MS46122A/B

6. Save the file by selecting the File option near the header and select Save As.
7. Name the file and push the Save button. Press F5 to run the script.
8. The input of the script should look like Figure 1-7.

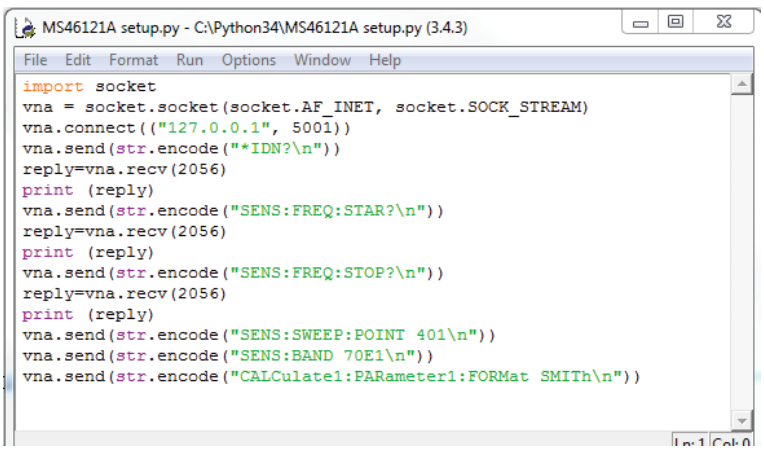


Figure 1-7. Execution of the MS46122A/B Python Script

Note	There are some extra commands that are needed to run a script that are not needed when inputting line by line commands in the Python Shell. A line termination is required after each string which is \n.

Connecting the MS46322A/B to Python 3.4 with a controller PC and Ethernet interface

The MS46322A/B can be run from the embedded PC or through a controller PC. This example will be done with an external controller PC using Python 3.4, PyVISA and a MS46322A/B. The controller PC will use an RJ-45 cable (Ethernet interface) to communicate with the MS46322A/B.

Note	PyVISA must be enabled on the controller PC. There are several resource available to get PyVISA and many online tutorials for loading the software.

1. Download Python 3.4 or the latest edition of Python 3.4 from the Python.org website: https://www.python.org/downloads/
2. In the section for downloads, choose the latest Python edition. i.e. Python 3.4.X
3. Run software and install for all users
4. Open the Windows icon on the PC's desktop and select All Programs
5. Find the folder labeled, "Python 3.4" and open this folder. When this file is open, choose IDLE (Python 3.4 GUI)
6. The Python 3.4.X shell will open and take line by line commands. See Figure 1-8

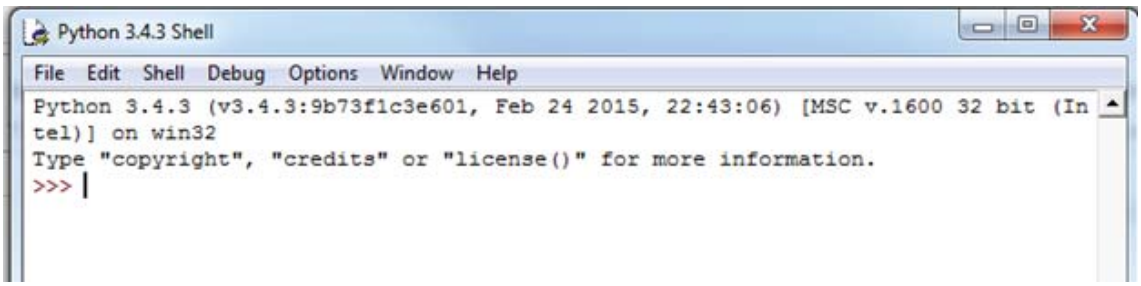


Figure 1-8. Python Shell

7. Connect the controller PC to the MS46322A/B.

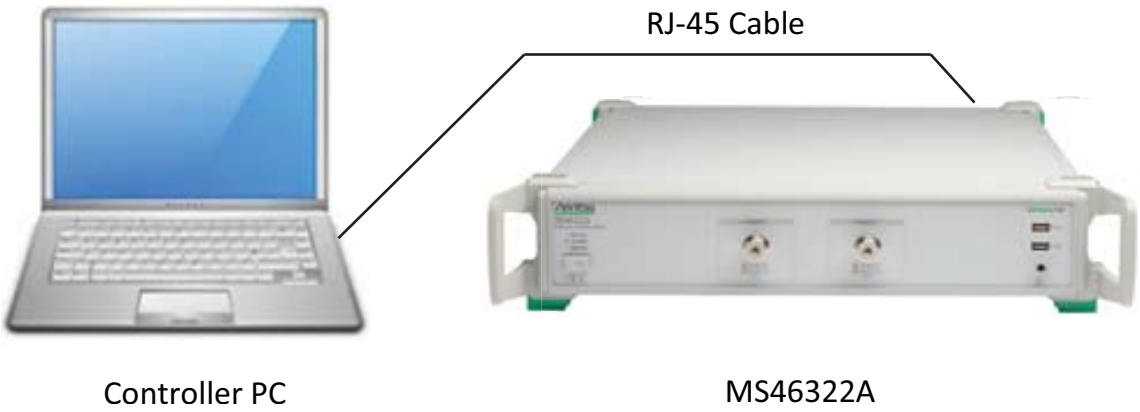


Figure 1-9. PC Controller to MS46322A/B Connection

8. Open the ShockLine application to run SCPI commands.

Note	If application is not launched before running Python, the following error will occur: No connection could be made because the target machine actively refused it.

9. Navigate to the IP Address in the Network Interface menu. SeeFigure 1-10.

Navigation:

Utilities | System | SYSTEM | Network Interface | NETWORK INTERFACE IP Address

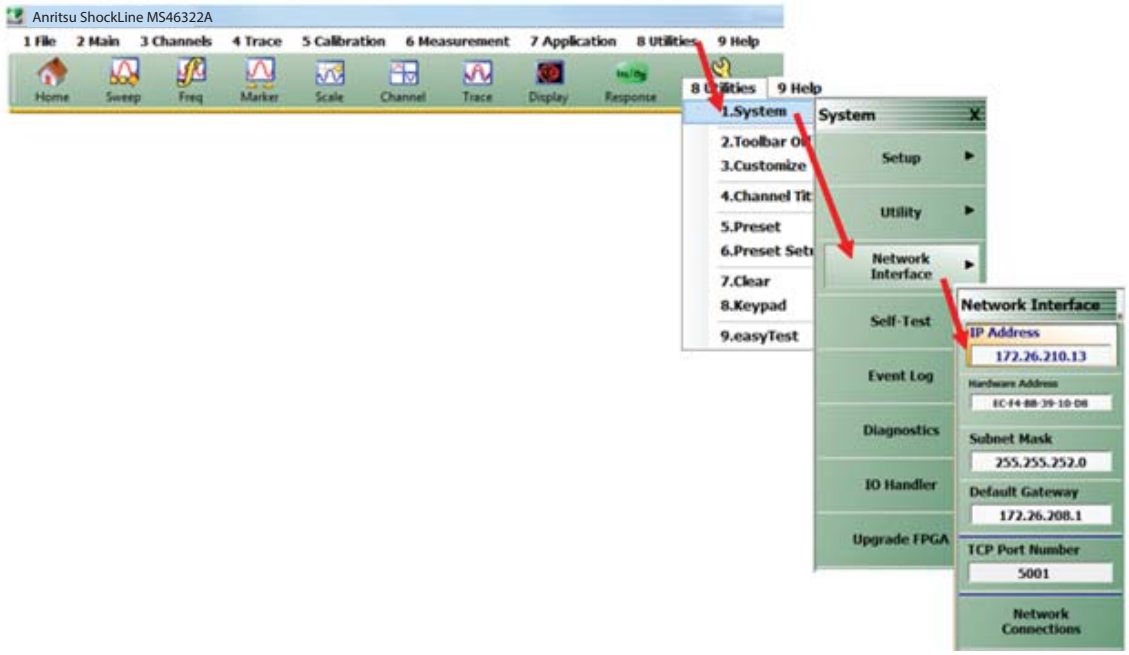


Figure 1-10. IP Address Menu Navigation Path

10. Type the following commands (CAPS matter). Press “Enter” after each command. IP Address will be found using Step 9.

- import visa
- rm=visa.ResourceManager()
- MS46322A/B=rm.open_resource('TCPIP0::IP Address::INSTR')
- print(MS46322A/B.query("*IDN?"))

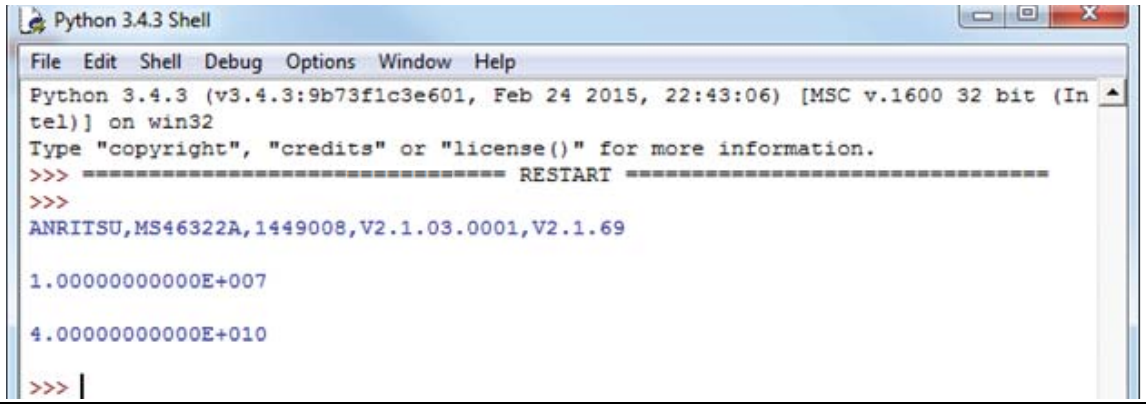


Figure 1-11. Instrument Identification Inquiry

Creating a Script using an MS46322A/B using Python 3.4 and PyVISA

This script will query the instruments general information along with the start frequency and stop frequencies. It will then set the number of points to 401 and the IFBW to 10 kHz and set trace 4 to Smith Chart.

1. Open the ShockLine application.
2. Open Python 3.4.3 Shell. This is labeled Python IDLE in the Python 3.4 folder.
3. In the Python Shell, choose File option near the header and select New File.
   - An Untitled file will open and it will be blank.
4. Type in the following commands:
   - import visa
   - rm=visa.ResourceManager
   - MS46322B=rm.open_resource
   - print(MS46322B.query("*IDN?"))
   - print (MS46322B.query("SENS:FREQ:STAR?"))
   - print (MS46322B.query("SENS:FREQ:STOP?"))
   - MS46322B.write("SENS:SWEEP:POINT 401")
   - MS46322B.write("SENS:BAND 70E1")
   - MS46322B.write("CALCulate1:PARAmeter4:FORMat SMITH")

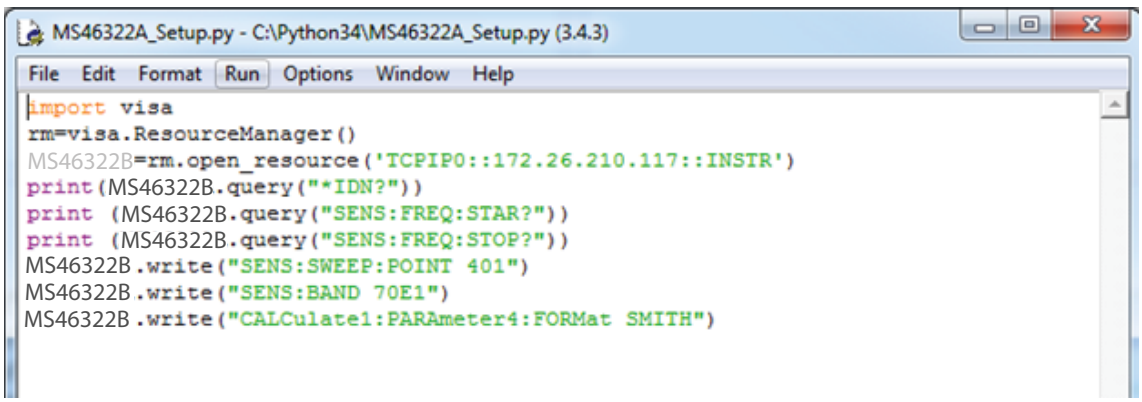


Figure 1-12. Python Script with SCPI Commands

5. Save the file by selecting the File option near the header and select Save As.
6. Name the file and push the Save button. Press F5 to run the script.
7. The input of the script should look like Figure 1-13.

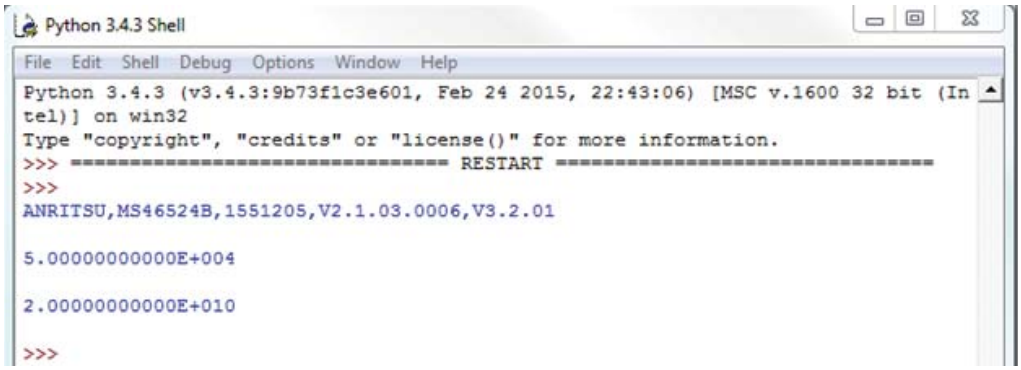


Figure 1-13. Execution of Python Script

Document Conventions

Note	Many of the images in this document are typical representations of the product or its features. Your instrument and its displays may vary slightly from these images.

User Interface Navigation

Elements in navigation shortcuts or paths are separated with the pipe symbol ("|"). Menu and dialog box names are distinctive Sans Serif font in CAPITALS. Button names are in Title Case. For example, the path to the Manual Cal menu is:

- MAIN | Calibration | CALIBRATION | Calibrate | CALIBRATE | Manual Cal | MANUAL CAL

1-6 Minimum/Maximum Instrument Frequency and Related Parameters

The minimum and maximum instrument frequencies depend on the instrument model and the installed options. The general frequency limits for the :SENSe{1-16}:FREQuency subsystem and related commands are defined in the following material.

Model MS46121A/B, MS46122A/B, MS46322A/B Standalone VNA Frequency Limits

The tables below provide standalone VNA frequency limits.

Standalone VNAs – Default Start, Default CW, and Default Stop Frequencies

The instrument default start and stop frequencies are listed below:

Table 1-2. Standalone VNAs – Default Start, Default CW, and Default Stop Frequencies

Models	Default Start or CW Frequency	Default Stop Frequencies
MS46121A/B-004	4.0000000 E+07 (40 MHz)	4.000000000 E+09 (4 GHz)
MS46121A/B-006	0.1500000 E+06 (150 kHz)	6.000000000 E+09 (6 GHz)
MS46122A/B-010	1.0000000 E+07 (10 MHz)	8.000000000 E+09 (8 GHz)
MS46122A/B-020	1.0000000 E+07 (10 MHz)	2.000000000 E+10 (20 GHz)
MS46122A/B-040	1.0000000 E+07 (10 MHz)	4.000000000 E+10 (40 GHz)
MS46322A-004	1.0000000 E+07 (10 MHz)	4.000000000 E+09 (4 GHz)
MS46322A/B-010	1.0000000 E+07 (10 MHz)	8.000000000 E+09 (8 GHz)
MS46322A-014	1.0000000 E+07 (10 MHz)	1.400000000 E+10 (14 GHz)
MS46322A/B-020	1.0000000 E+07 (10 MHz)	2.000000000 E+10 (20 GHz)
MS46322A-030	1.0000000 E+07 (10 MHz)	3.000000000 E+10 (30 GHz)
MS46322A/B-040	1.0000000 E+07 (10 MHz)	4.350000000 E+10 (43.5 GHz)

Standalone VNAs – Minimum Start, Minimum CW, and Maximum Start Frequencies

The highest possible setting for the Start Frequency is the Stop Frequency minus 2 Hz. This yields a sweep with three data points.

Table 1-3. Standalone VNAs – Minimum Start, Minimum CW, and Maximum Start Frequencies

Model	Minimum Start or CW Frequency	Maximum Start Frequency (Stop Frequency – 2 Hz)
MS46121A/B-004	4.0000000 E+07 (40 MHz)	3.999999998 E+09 (4GHz - 2 Hz)
MS46121A/B-006	0.1500000 E+06 (150 kHz)	5.999999998 E+09 (6 GHz - 2 Hz)
MS46122A/B-010	1.000000 E+06 (1 MHz)	7.999999998 E+09 (8 GHz - 2 Hz)
MS46122A/B-020	1.000000 E+06 (1 MHz)	1.9999999998 E+10 (20 GHz - 2 Hz)
MS46122A/B-040	1.000000 E+06 (1 MHz)	4.3499999998 E+10 (43.5 GHz - 2 Hz)
MS46322A-004	1.000000 E+06 (1 MHz)	3.999999998 E+09 (4 GHz - 2 Hz)
MS46322A/B-010	1.000000 E+06 (1 MHz)	7.999999998 E+09 (8 GHz - 2 Hz)
MS46322A-014	1.000000 E+06 (1 MHz)	1.3999999998 E+10 (14 GHz - 2 Hz)
MS46322A/B-020	1.000000 E+06 (1 MHz)	1.9999999998 E+10 (20 GHz - 2 Hz)
MS46322A-030	1.000000 E+06 (1 MHz)	2.9999999998 E+10 (30 GHz - 2 Hz)
MS46322A/B-040	1.000000 E+06 (1 MHz)	4.3499999998 E+10 (43.5 GHz - 2 Hz)

Standalone VNAs – Minimum Stop and Maximum Stop Frequencies

The lowest possible setting for the stop frequency is the start frequency plus 2 Hz which yields as sweep of three (3) data points.

Table 1-4. Standalone VNAs – Minimum Stop, Maximum Stop, and Maximum CW Frequencies

Model	Minimum Stop Frequency Stop Min = Start + 2 Hz = 3 data points	Maximum Stop Frequency or Maximum CW Frequency Stop Max = Instrument Max)
MS46121A/B-004	4.0000002 E+07 (40 MHz + 2 Hz)	4.000000000 E+09 (4 GHz)
MS46121A/B-006	1.50002 E+05 (150 kHz + 2 Hz)	6.000000000 E+09 (6 GHz)
MS46122A/B-010	1.000002 E+06 (1 MHz + 2 Hz)	8.000000000 E+09 (8 GHz)
MS46122A/B-020	1.000002 E+06 (1 MHz + 2 Hz)	2.000000000 E+10 (20 GHz)
MS46122A/B-040	1.000002 E+06 (1 MHz + 2 Hz)	4.350000000 E+10 (40 GHz)
MS46322A-004	1.000002 E+06 (1 MHz + 2 Hz)	4.000000000 E+09 (4 GHz)
MS46322A/B-010	1.000002 E+06 (1 MHz + 2 Hz)	8.000000000 E+09 (8 GHz)
MS46322A-014	1.000002 E+06 (1 MHz + 2 Hz)	1.400000000 E+10 (14 GHz)
MS46322A/B-020	1.000002 E+06 (1 MHz + 2 Hz)	2.000000000 E+10 (20 GHz)
MS46322A-030	1.000002 E+06 (1 MHz + 2 Hz)	3.000000000 E+10 (30 GHz)
MS46322A/B-040	1.000002 E+06 (1 MHz + 2 Hz)	4.350000000 E+10 (40 GHz)

Standalone VNAs – Default Frequency Span and Maximum Frequency Span

The frequency span equals the stop frequency minus the start frequency. The minimum possible frequency span is frequency is 2 Hz.

Table 1-5. Standalone VNAs – Default Frequency Span and Maximum Frequency Span

Model	Default Frequency Span	Maximum Frequency Span Span Max = Stop – Start
MS46121A/B-004	3.960000000 E+09 (4 GHz - 40 MHz)	3.960000000 E+09 (4 GHz - 40 MHz)
MS46121A/B-006	5.999850000 E+09 (6 GHz - 150 kHz)	5.999850000 E+09 (6 GHz - 150 kHz)
MS46122A/B-010	7.990000000 E+09 (8 GHz - 10 MHz)	7.999000000 E+09 (8 GHz - 10 MHz)
MS46122A/B-020	1.999000000 E+10 (20 GHz - 10 MHz)	1.999900000 E+10 (20 GHz - 10 MHz)
MS46122A/B-040	3.999000000 E+10 (40 GHz - 10 MHz)	4.349900000 E+10 (43.5 GHz - 10 MHz)
MS46322A-004	3.990000000 E+09 (4 GHz - 10 MHz)	3.999000000 E+09 (4 GHz - 10 MHz)
MS46322A/B-010	7.990000000 E+09 (8 GHz - 10 MHz)	7.999000000 E+09 (8 GHz - 10 MHz)
MS46322A-014	1.399000000 E+10 (14 GHz - 10 MHz)	1.399900000 E+10 (14 GHz - 10 MHz)
MS46322A/B-020	1.999000000 E+10 (20 GHz - 10 MHz)	1.999900000 E+10 (20 GHz - 10 MHz)
MS46322A-030	2.999000000 E+10 (30 GHz - 10 MHz)	2.999900000 E+10 (30 GHz - 10 MHz)
MS46322A/B-040	3.999000000 E+10 (40 GHz - 10 MHz)	4.349900000 E+10 (43.5 GHz - 10 MHz)

Standalone VNAs – Minimum Center and Maximum Center Frequencies

The minimum possible center frequency is the minimum start frequency plus 1 Hz. The maximum possible center frequency is the maximum stop frequency minus 1 Hz.

Table 1-6. Standalone VNAs – Minimum Center Frequency and Maximum Center Frequency

Model	Minimum Center Frequency Center Min = Start + 1 Hz	Maximum Center Frequency All Models Center Max = Stop – 1 Hz
MS46121A/B-004	4.0000001 E+07 (40 MHz + 1 Hz)	3.999999999 E+09 (4 GHz - 1 Hz)
MS46121A/B-006	1.50001 E+05 (150 kHz + 1 Hz)	5.999999999 E+09 (6 GHz - 1 Hz)
MS46122A/B-010	1.000001 E+06 (1 MHz + 1 Hz)	7.9999999990 E+09 (8 GHz - 1 Hz)
MS46122A/B-020	1.000001 E+06 (1 MHz + 1 Hz)	1.9999999999 E+10 (20 GHz - 1 Hz)
MS46122A/B-040	1.000001 E+06 (1 MHz + 1 Hz)	4.3499999998 E+10 (43.5 GHz - 1 Hz)
MS46322A-004	1.000001 E+06 (1 MHz + 1 Hz)	3.9999999990 E+09 (4 GHz - 1 Hz)
MS46322A/B-010	1.000001 E+06 (1 MHz + 1 Hz)	7.9999999990 E+09 (8 GHz - 1 Hz)
MS46322A-014	1.000001 E+06 (1 MHz + 1 Hz)	1.3999999999 E+10 (14 GHz - 1 Hz)
MS46322A/B-020	1.000001 E+06 (1 MHz + 1 Hz)	1.9999999999 E+10 (20 GHz - 1 Hz)
MS46322A-030	1.000001 E+06 (1 MHz + 1 Hz)	2.9999999999 E+10 (30 GHz - 1 Hz)
MS46322A/B-040	1.000001 E+06 (1 MHz + 1 Hz)	4.3499999998 E+10 (43.5 GHz - 1 Hz)

Standalone VNAs – Default Center Frequencies

The center frequency is equal to Start Frequency plus the Stop Frequency divided by two (2). The minimum possible frequency span is 2 Hz.

Table 1-7. Standalone VNAs – Default Center Frequencies

Model	Default Center Frequency Center Default = (Start + Stop)/2 Center Minimum Frequency Span = 2 Hz
MS46121A/B-004	2.020000000 E+09
MS46121A/B-006	3.000075000 E+09
MS46122A/B-010	4.005000000 E+09
MS46122A/B-020	1.000500000 E+10
MS46122A/B-040	2.000500000 E+10
MS46322A-004	2.005000000 E+09
MS46322A/B-010	4.005000000 E+09
MS46322A-014	7.005000000 E+09
MS46322A/B-020	1.000500000 E+10
MS46322A-030	1.500500000 E+10
MS46322A/B-040	2.000500000 E+10

Chapter 2 — Programming the ShockLine™ Series VNA

2-1 Introduction

This chapter provides an introduction to programming the ShockLine VNA with the SCPI programming language. It also includes descriptions of the command types the instrument accepts, program command structures, data parameters and input/output specifications, and notational conventions. Information on the instrument's status system and trigger system programming is also provided.

Note	When the ShockLine VNA is operated through remote programming, the ShockLine™ application screen's user interface controls are disabled (grayed out). To return to local screen control, press the keyboard **Esc** key, or send the RTL command.

2-2 Introduction to SCPI Programming

The Standard Commands for Programmable Instruments (SCPI) protocol defines a set of standard programming commands for use by all SCPI compatible instruments. SCPI is intended to give the ATE user a consistent environment for program development. It does so by defining controller messages, instrument responses, and message formats for all SCPI compatible instruments. The IEEE 488 interface for the VNA is designed to conform to the requirements of SCPI 1999.0. The set of SCPI commands implemented by the instrument interface provides a comprehensive set of programming functions covering all of the major functions of the instrument.

Command Types

SCPI commands, which are also referred to as SCPI instructions, are messages to the instrument to perform specific tasks. The instrument's command set, introduced in this chapter, includes these command types:

- "IEEE 488.2 Commands"
- "System Commands"
- "SCPI Commands"
- "Native SCPI Commands"

2-3 IEEE 488.2 Commands

The IEEE-488.2 commands are defined in the IEEE-488.2 standard and must be implemented by all SCPI compatible instruments. The mandated commands listed in Table 2-1 are identified by the asterisk (*) at the beginning of the command keyword. These commands are used to control instrument status registers, status reporting, synchronization, and other common functions. The IEEE 488.2 required common commands are described in detail in the first half of Chapter 3, "IEEE Commands" starting with "IEEE 488.2 Commands" on page 3-2.

Table 2-1. IEEE 488.2 Mandated Commands

*CLS	*ESE	*IDN?	*OPT?	*SRE?
*DDT	*ESE?	*OPC	*RST	*STB?
*DDT?	*ESR?	*OPC?	*SRE	*TRG
				*TST?
				*WAI

2-4 System Commands

The set of system commands are primarily used to control the state of the instrument for system diagnostics, hardware calibration, and troubleshooting.

2-5 SCPI Commands

There are two general classifications of SCPI commands described in the two sections below:

- Required (or mandated) SCPI Commands
- Native SCPI Commands

Note that the Required SCPI Commands are a subset of the Native SCPI commands.

Required SCPI Commands

The required SCPI commands are listed in the table below:

Table 2-2. SCPI Required or Mandated Commands

:STATus	:SYSTem
:OPERation	:ERRor
[:EVENt]?	[:NEXT]?
:CONDition?	
:ENABle	
:QUEStionable	
[:EVENt]?	
:CONDition?	
:ENABle	

Native SCPI Commands

The majority of the commands are native SCPI commands and are also described in detail in Chapter 5, "SCPI Commands: 1-Port and 2-Port VNAs". Depending on the number of keywords in the command, the subsystems are grouped by either the first two keywords (such as :CALCulate{1-16}:MARKer Subsystem) or the first three keywords (such as :CALCulate{1-16}[:SELected]:CONVersion Subsystem).

The commands are listed in strict ASCII sort sequence.

See the sections below starting with "Command Requirements" on page 2-3 for definitions of parameters and other notations.

2-6 Command Requirements

Query Commands

All commands, unless specifically noted in the commands syntax descriptions, have a query form. Exceptions are noted as:

- Commands without a query form have a query status of "No Query".
- Queries without a command form have a query status of "Query Only".

As defined in IEEE-488.2, a query is a command with a question mark symbol appended (examples are `*ESR?` and `*TST?`). When a query form of a command is received, the current setting associated with the command is placed in the output buffer. Query commands always return the short form of the parameter. For example, `NORMal` or `INVerted` is returned as `NORM` or `INV`. Boolean values are returned as `1` or `0`, even when they can be set as `ON` or `OFF`.

Command Names

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

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

Some command keywords may have a numeric suffix to differentiate between multiple instrument features such as multiple pulse widths. For example, keywords `:WIDTh2` (or `:WIDT2`).

As with any programming language, the exact command keywords and command syntax must be used. The syntax of the individual commands is described in detail in Chapter 5, "SCPI Commands: 1-Port and 2-Port VNAs". Unrecognized versions of long form or short form commands, or improper syntax, will generate an error.

Hierarchical Command Structure

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

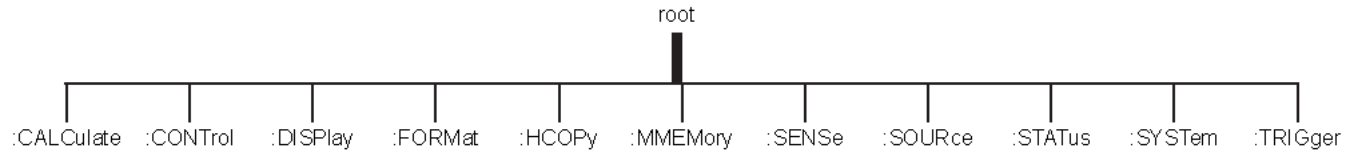


Figure 2-1. ShockLine™ VNA Partial SCPI Command Tree

All ShockLine™ VNA SCPI commands have one or more subcommands (keywords) associated with them to further define the instrument function to be controlled. The subcommand keywords may also have one or more associated subcommands (keywords). Each subcommand level adds another layer to the command tree. The command keyword and its associated subcommand keywords form a portion of the command tree called a command subsystem. The `:FORMat` command subsystem is shown below.

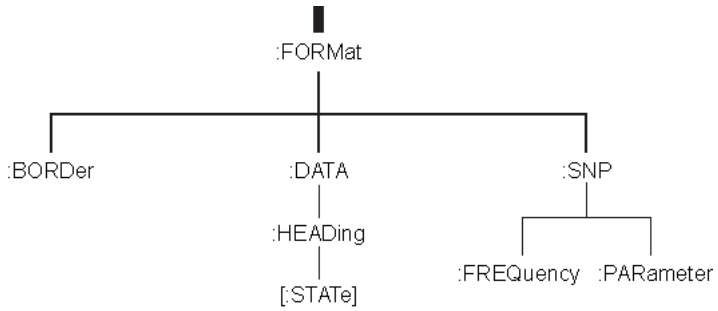


Figure 2-2. SCPI `:FORMat` Subsystem

Data Parameters

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

The command descriptions in Chapter 5, "SCPI Commands: 1-Port and 2-Port VNAs" specify the type of data parameter to be used with each command. The most commonly used parameter types are numeric, extended numeric, discrete, and Boolean.

- **Numeric**

  Numeric parameters comprise integer numbers or any number in decimal or scientific notation, and may include polarity signs. This includes <NR1>, <NR2>, and <NR3> numeric data as defined in "Parameter Notations" on page 2-6. This type of numeric element is abbreviated as <NRf> throughout this document.

- **Discrete**

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

- **Boolean**

  Boolean parameters represent binary conditions and may be expressed as ON, OFF or 1, 0.

Note	The ShockLine command parser will generally accept all numerical values within the parameter ranges specified. In cases where a command parameter value is outside of the indicated range or resolution of the instrument, the nearest appropriate value will be entered.

2-7 Notational Conventions

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

General Notations

The syntax conventions that are used for all SCPI command keywords and data parameter descriptions in this manual are described below:

Table 2-3. General Notations

:	A colon links command keywords together to form commands. The colon is not an actual part of the keyword, but is a signal to the SCPI interface parser. A colon must precede a root keyword immediately following a semicolon (see “Notational Examples” on page 2-7).
;	A semicolon separates commands if multiple commands are placed on a single program line (see “Notational Examples” on page 2-7).
[ ]	Square brackets enclose one or more optional keywords.
{ }	Braces enclose one or more keyword parameters that may be included one or more times.
|	A vertical bar (also called a “pipe”) indicates “or” and is used to separate alternative parameter options. For Example: ON | OFF is the same as ON or OFF.
< >	Angle brackets enclose parameter descriptions.
::=	Means “is defined as” For example: <a>::=<b><c> indicates that <b><c> can replace <a>.

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

Parameter Notations

The following syntax conventions are used for all data parameter descriptions in this manual:

Table 2-4. Parameter Notations

Parameter	Definition
<ASCII>	A non-delimited 7-bit ASCII text. The end of the text must be terminated with the 0A character (decimal 10) and concurrent setting (^) of the GPIB End of Transmission State (EOI). <ASCII> (also called <Arbitrary ASCII>) text is transmitted only at the end of a program or response message.
<block>	IEEE-488.2 block data format. Can be in ASCII, XML, or other format.
<bNR1>	Boolean values in <NR1> format; numeric 1 or 0
<boolean>	ON | OFF. Can also be represented as 1 or 0, where 1 means ON and 0 means OFF Boolean parameters are always returned as 1 or 0 in <NR1> format by query commands
<char>	<CHARACTER PROGRAM DATA> Examples: CW, FIXed, UP, and DOWN
<INF>	Positive Infinity. Positive infinity is represented as 9.9E37. The numeric value for positive infinity fits into a 32-bit IEEE 754 floating point number.
<integer>	An unsigned integer without a decimal point (implied radix point)
<NA>	Not Applicable
<NAN>	Not A Number. Not a number is represented as 9.91E37 and is defined in IEEE 754. Typically used where applications are dividing zero by zero or subtracting infinity from infinity. NAN is also used to represent missing data such as a trace that has not been yet acquired.
<NINF>	Negative Infinity. Negative infinity is represented as -9.9E37. The numeric value for negative infinity fits into a 32-bit IEEE 754 floating point number.
<NR1>	A signed integer without a decimal point (implied radix point)
<NR2>	A signed number with an explicit radix point
<NR3>	A scaled explicit decimal point numeric value with an exponent (e.g., floating point number)
<NRf>	Values in NR1, NR2, or NR3 formats are accepted. Logically, <NR1> | <NR2> | <NR3>
<string>	<STRING PROGRAM DATA> ASCII characters surrounded by double quotes For example: “C:\Anritsu\ShockLine\filename.s2p”
MPND	Numeric Limit. Maximum Positive/Negative Double Precision Number. +/– 1.792 693 134 860 E+308
MPNF	Numeric Limit. Maximum Positive/Negative Float Number +/- 3.402 819 E+38
MPNI	Numeric Limit. Maximum Positive/Negative Integer - 2 147 483 648 to +2 147 483 647

Refer to “Data Transmission Methods” on page 2-9 for detailed information about parameter input/output and transferring data to/from the instrument.

Notational Examples

The following is an example showing command syntax:

```
:SENSe{1-16}:FREQuency:STARt 2.0E9
```

Command statements read from left to right and from top to bottom. In the command statement above, the `:FREQuency` keyword immediately follows the `:SENSe{1-16}` keyword with no separating space. The braces { } indicate an optional keyword parameter.

For commands in general, if the keyword's optional parameter is not used, a value of 1 is assumed. However, for commands involving a channel number, if no value is provided, the command applies to ALL channels.

A space is required between the command string and its argument.

Note that the first keyword in the command string does not require a leading colon; however, it is good practice to always use a leading colon for all keywords.

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

```
:SENSe3:FREQuency:STARt 2.0E9;:SENSe3:FREQuency:STOP 20.0E9
```

Using the command keyword short form, the command string above would be:

```
:SENS3:FREQ:STAR 2.0E9;:SENS3:FREQ:STOP 20.0E9
```

Note the semicolon used to join the commands. Also note the leading colon used immediately after the semicolon.

2-8 Numeric Data Suffix Reference

Unit suffixes are not required for data parameters, provided the values are scaled for the global default units. The VNA's SCPI default units are:

- Hz (hertz) for frequency-related parameters
- s (second) for time-related parameters
- m (meter) for distance-related parameters
- ohm for impedance-related parameters
- dB for power-related parameters
- Henry and Farad for reactance-related parameters

For example, the commands below, move the VNA starting marker frequency to 3 GHz.

```
:CALC:MARK:MOV:STAR 3000000000
:CALC:MARK:MOV:STAR 3.0 E9
```

The following table provides a reference to the I/O parameter types (and the appropriate multiplier) used with the VNA.

Table 2-5. Numeric Data Suffix

Code	Parameter Type	Multiplier
DB, DBL, DBM	Power	1.0
DEG	Phase	1.0
RAD	Phase	180/π (180/Pi)
HZ	Frequency (Hertz)	1.0
KHZ	Frequency (Kilohertz)	1.0E3
MHZ	Frequency (Megahertz)	1.0E6
GHZ	Frequency (Gigahertz)	1.0E9
REU	Real	1.0
IMU	Imaginary	1.0
S	Time	1.0
MS	Time (Millisecond)	1.0E-3
US, USC	Time (Microsecond)	1.0E-6
NS, NSC	Time (Nanosecond)	1.0E-9
PS, PSC	Time (Picosecond)	1.0E-12
M, MTR	Distance (Meter)	1.0
CM, CMT	Distance (Centimeter)	1.0E-2
MM, MMT	Distance (Millimeter)	1.0E-3
OHM	Impedance	1.0
V, VLT	Voltage	1.0
MV	Voltage (Millivolt)	1.0E-3
XM3	Unitless	1.0E-3
XX1	Unitless	1.0
XX3	Unitless	1.0E3

2-9 Data Transmission Methods

Data transmissions to and from the VNA conform to the protocols specified by the IEEE 488.2 Standard. The 488.2 Standard specifies how any data, such as ASCII numbers, strings, or blocks of data bytes, will be transmitted over the Ethernet. This section describes the various transmission methods in use by the VNA. The transmission method names described below (also called notations) will be used throughout the Programming Manual when describing specific VNA data transfer commands. Data transmission notations are easily distinguished in text as they are always shown surrounded by the “less than” and the “greater than” characters (< >). The transmission type notations used in describing various VNA data transmissions are:

- For ASCII numbers, the notations are: <NR1>, <NR2>, <NR3>, or <NRf>
- For ASCII strings (printable characters and print formatting codes), the notation is: <string>
- For generic (7-bit) ASCII characters, the notation is: <Arbitrary ASCII>
- For generic binary bytes, (7-bit ASCII or binary), the notation is: <block>

<NR1>

This notation represents ASCII integer values. A comma (,) is used to separate multiple values sent in a single command's input or output string. Examples of values that can be represented by <NR1> notation:

```
10
-29,179
```

<NR2>

This notation represents ASCII floating point values in decimal point format. A comma (,) is used to separate multiple values sent in a single command's input or output string. Examples of values that can be represented by <NR2> notation:

```
1.0
-0.00015
12.743,-180.07
```

<NR3>

This notation represents ASCII floating point values in exponential format (scientific notation). A comma (,) is used to separate multiple values sent in a single command's input or output string. Examples of values that can be represented by <NR3> notation:

```
1.0E9
-7.056E3
9.0E-2,3.42E2
```

<NRf>

This notation is used to signify that data can be in either <NR1>, <NR2>, or <NR3> format as described above. Examples of values that can be represented by <NRf> notation:

```
1.0E-9
10.005
-83,4.5E2,-234.9901
```

<string>

This notation represents a string of ASCII characters (including non-printable characters) that is delimited (surrounded) with either single quotes (' ') or double quotes (“ “).The string can include text formatting characters such as line feed, space, carriage return, or printer control characters. Note that if a double quote character must be sent as part of the string, then it must be followed by an additional double quote. Alternatively, the string can be sent using single quotes (See "cal file" example below). Examples of data represented by <string> notation:

```
"1/15/98"
"Save ""cal_file"" now"
'Save "cal_file" now'
```

<ASCII> or <Arbitrary ASCII>

This notation represents undelimited 7-bit ASCII text. The end of the text must be terminated with the 0A character (decimal 10) and concurrent setting (^) of the GPIB End of Transmission State (EOI). This requirement makes it necessary for <Arbitrary ASCII> text to be transmitted only at the end of a program or response message, i.e., at the end of a multiple input or output statement. Example of data represented by <Arbitrary ASCII> notation:

```
ANRITSU,MS46322A/B,123456,1.0<0A^EOI>
```

The example shows a sample response from the *IDN?, 488.2 common query. In the example, the instrument identifies itself as an ANRITSU MS46322A/B, with serial number 123456, and software version 1.0 installed.

<block> or <arbitrary block>

This notation represents data that is transmitted as 8-bit data bytes (00–FF hex, 0–255 decimal, notation is <DAB>). This is useful for transmitting large blocks of:

- Formatted ASCII data
- Formatted XML data
- Formatted binary data
- Unformatted binary data

The data stream is immediately preceded by a variable length ASCII header that is encoded with the number of data bytes to be sent. The header always starts with the pound (#) character. The header and the transmitted data messages are described as follows:

```
#nm1..mn<DAB1>...<DABm>
```

Where:

= The pound sign character. Required for binary data transfer.

n = Number of digits to follow (m1..mn) that make up the number m.

m1..mn = Taken together, this makes up the number m which is the number of data bytes to follow that constitute the requested data.

<DAB> = An 8 bit binary data byte. This is the data (or information) being sent.

Note	If n = 0, then m is omitted, and transmission end is signaled by sending the line feed character (0A, or decimal 10) and concurrent setting (^) of the End Of Transmission State (EOI) immediately following the last <DAB>.

Example 1: #3204<DAB1>...<DAB204>

Example 1 shows how 204 8-bit bytes are transmitted using the proper header. The header in this example is comprised of 5 characters (#3204). It begins with the pound character (#). The next character (3) indicates there are 3 digits to follow that indicate the number of bytes being transmitted (204). The next three characters (204) indicate the number of data bytes being transmitted immediately after the header. Next comes the actual data bytes, or information, being transmitted (<DAB1>...<DAB204>).

Example 2: #512808<DAB1>...<DAB12808>

Example 2 shows how 12808 bytes are transmitted using the proper header. The header in this example is comprised of 7 characters (#512808). It begins with the pound character (#). The next character (5) indicates there are 5 digits to follow that indicate the number of bytes being transmitted (12808). The next five characters (12808) indicate the number of data bytes being transmitted immediately after the header. Next comes the actual data bytes, or information, being transmitted (<DAB1>...<DAB12808>).

Note	Examples 1 and 2 above demonstrate the <block> form referred to as <Definite Length Arbitrary Block>. It is so called because the number of data bytes being transmitted is known from the encoded header.

Example 3: #0<DAB1>...<DABm><0A^EOI>

Example 3 shows how an unknown number of bytes are transmitted using the proper header. The header in this example is comprised of 2 characters (#0). As usual, the header begins with the pound character (#). The next character (0) indicates there is an unknown number of data bytes being transmitted immediately after the header. Next comes the actual data bytes being transmitted (<DAB1>...<DABm>). The end of the data stream is signaled by sending the line feed character (0A, or decimal 10) and concurrent setting (^) of the End of Transmission State (EOI).

Note	Example 3, above, demonstrates a special form of <block> data referred to as the <Indefinite Length Arbitrary Block>. It is so called because the number of data bytes being transmitted is unknown, and therefore cannot be encoded in the header. Instead, the header always consists of the pound and zero characters (#0) and end of the data stream is always signaled by sending the line feed character (0A, or decimal 10) and concurrent setting (^) of the End of Transmission State (EOI). This requirement makes it necessary for <Indefinite Length Arbitrary Block> text to be transmitted only at the end of a program or response message (at the end of a multiple input or output statement).

<char>

Character program data such as CW, FIXed, UP, and DOWN. A single instance in a command or query is <char>. If multiple instances are required, each is identified such as <char1> or <char2> and the individual elements are separated by commas:

- <char1>,<char2>
- <char1>,<char2>,<char3>
- <char1>,<char2>,<char3>,<char4>

MPND

The instrument numeric limit as the Maximum Positive/Negative Double Precision Number or:

```
+/– 1.792 639 134 86 E+308
```

MPNF

The instrument numeric limit as the Maximum Positive/Negative Float Number or:

```
+/- 3.402 819 E+38
```

MPNI

The instrument numeric limit as the Maximum Positive/Negative Integer or:

```
- 2 147 483 648 to +2 147 483 647
```

Formatting Data Output

Three commands are provided to alter the way the arbitrary block header for output data is formed.

- **FDH0**

  Specifies that the length of the arbitrary block header will be minimized; that is, the byte count section will not contain leading zeros, thus its length is indeterminate. This means that a program must decode the header in order to skip over it.

- **FDH1**

  Specifies that the length of the arbitrary block header will be fixed at 11 characters. This is accomplished by forcing leading zeros as required in the byte count section. This means that a program can skip over the arbitrary block header by skipping 11 characters. FDH1 is the default mode.

- **FDH2**

  Specifies that no arbitrary block header will be sent with the next transmission. This mode is not in compliance with IEEE 488.2 specifications and will persist for all subsequent program messages.

- **FDHX?**

  FDH mode query yields the following results:

  - 0: FDH0
  - 1: FDH1
  - 2: FDH2

ASCII or Binary Data Format

The following sections discuss the various data output formats:

Non-Array Data

The formats used for data transfers not involving numerical data arrays are preset. They always occur in either binary format or ASCII format, depending on the data. These data transfers include a variety of information. Examples include:

- Instrument setup strings
- Marker data, queries
- Disk directory listings

Numerical Data Arrays

Numerical data array transfers are used to transfer the following types of data:

- Measurement data
- Calibration data
- Sweep frequency, time, or distance values

Each of these data transfer types are individually explained below. You can select either binary or ASCII format for data transfers involving numerical data arrays. The commands described below select and keep the format for all subsequent data transfers.

- **ASCII Format - FMA**

  ASCII formatted values represented in <NR1>, <NR2>, <NR3>, or <NRf> formats. The VNA accepts any of the above formats as input. It will always output values using <NR3> exponential format with each value represented using 18 characters plus a comma to separate multiple values.

- **Binary Formats**
  - **FMB**

    Each eight consecutive data bytes represent one floating point value in IEEE 754 64-bit format (double precision, 8 byte, floating point value).

  - **FMC**

    Each four consecutive data bytes represent one floating point value in IEEE 754 32-bit format (single precision, 4 byte, floating point value).

  - **FMX?**

    FMA, FMB, FMC format selection query

- **Byte Ordering**
  - **MSB**

    Byte ordering is most significant byte first. For use only with FMB and FMC. This is the optional byte mode for the VNA.

  - **LSB**

    Byte ordering is least significant byte first. For use with FMB and FMC. This is required for transferring data to/from Intel/IBM based computers. LSB is the default mode.

- **XSB?**

  MSB, LSB format selection query.

- **FMT0**

  Turn ASCII enhancement off (normal default mode).

- **FMT1**

  Turn ASCII enhancement on.

- **FMTX?**

ASCII enhancement ON/OFF status query.

The following SCPI commands select either ASCII or Binary format as described above:

```
:FORMat:DATA <char>
```

Where the **<char>** arguments of ASC or ASCII ::= FMA, REAL ::= FMB, REAL32 ::= FMC

`:FORMat:DATA?` is the ASC or ASCII, REAL, or REAL32 format selection query.

```
:FORMat:BORDer <char>
```

Where the **<char>** arguments of NORMAL ::= MSB, SWAPPED ::= LSB

`:FORMat:BORDer?` is the MSB | LSB format selection query.

Enhanced ASCII Formatting

Enhanced ASCII formatting can be applied to both non-array ASCII data and numerical data arrays in the FMA format when this data is output within an <block> format. The format selectively replaces comma data element separators with a line feeds (ASCII 10) in order to enhance the visual effect. The following provides two examples of this enhanced structure:

- An unenhanced directory listing

```
#9000000392Directory of C:\ 1-30-96 13:03,UTIL <DIR> 1-25-96 12:58,PLOT
BMB 38462 1-22-96 14:41,PLOT BMC 307446 1-22-96 14:41,TTT CAL
44174 1-22-96 17:02,TTT2 CAL 44174 1-22-96 17:16,PLOT1 DAT
10323 1-22-96 14:03,PLOT1 HGL 19899 1-22-96 14:02,PLOT2 HGL
38462 1-25-96 13:16,8 Files 502940 Bytes
```

- An enhanced directory listing

```
#9000000392
Directory of C:\ 1-30-96 13:03
UTIL <DIR> 1-25-96 12:58
PLOT BMB 38462 1-22-96 14:41
PLOT BMC 307446 1-22-96 14:41
TTT CAL 44174 1-22-96 17:02
TTT2 CAL 44174 1-22-96 17:16
PLOT1 DAT 10323 1-22-96 14:03
PLOT1 HGL 19899 1-22-96 14:02
PLOT2 HGL 38462 1-25-96 13:16
8 Files 502940 Bytes
```

- An unenhanced response to OCD

```
#9000000189-9.99750733376E-01, 3.21409821510E-01, 3.60706359148E-01,
9.82860028744E-01, 7.76742696762E-01,-5.06587028503E-01,-5.07535457611E-01,
-8.45697641373E-01,-6.10321164131E-01,6.05827927589E-01
```

- An enhanced response to OCD

```
#9000000189
-9.99750733376E-01, 3.21409821510E-01
3.60706359148E-01, 9.82860028744E-01
7.76742696762E-01,-5.06587028503E-01
-5.07535457611E-01,-8.45697641373E-01
-6.10321164131E-01, 6.05827927589E-01
```

2-10 Calculating the Byte Size

This section describes the factors for calculating the byte size of responses to selected remote-only queries. The byte size of the resultant data from several of the remote only queries depends on several factors:

- Parameters per Output
- Numbers Output per Data Point
- Bytes Output per Number
- Size of Block Data
- Number of Bytes Output

Numbers Output-per-Data Point (NODP)

The data for each data point is a complex number (A + jB) where A and B are floating point numbers. This data is saved internally for use and possible future output. Additionally, if an RF correction is active, the RF correction is applied to the RAW measurement and the result is saved internally for use and possible future output. Either the RAW or CORRECTED data are taken and converted into the data format for the display type selected.

This data is saved internally in the FORMATTED (final) measurement form for use and possible future output. When this conversion takes place, the data will, in most cases, still be two orthogonal numbers. However, several of the displays types throw away a portion of the data and the result will be one number only. The display types that produce only one number are:

- Group Delay
- Imaginary
- Linear Magnitude
- Log Magnitude
- Phase
- Power Out
- Real
- SWR

To summarize, the RAW, CORRECTED, and FORMATTED data output will be two numbers-per-point, unless the display type is one of those mentioned above.

Bytes Output-per-Number (BOPN)

The number of bytes output per number is shown below:

Table 2-6. Bytes Output per Number

Number	Output Format	Output-per-Number
FMA	(ASCII)	14 plus comma (short form data) 19 plus comma (long form data)
FMB	(double precision binary)	8
FMC	(single precision binary)	4

Size of Data Block (SODB)

In the case where there is only one parameter to output, the formula is:

```
SODB = NODP * BOPN * Number of points in the sweep
```

If the command is O4SC, O4FD, or O4SR, the formula is:

```
SODB = 8 * BOPN * Number of points in the sweep
```

Number of Bytes Output (NBO)

The number of bytes output is the number of bytes transmitted over the GPIB. In most cases, the data block is proceeded by an arbitrary block header followed by an end character (line feed), as shown below:

- Response Message = [Arbitrary Block Header] + [Data Block] + [End Character]

The size of the end character is one byte. The size of the arbitrary block header is variable between 2 and 11. If we always assume an arbitrary block header size of 11, then: NBO = 12 + SODB. For example:

- The VNA is set up for a one channel, four-trace display with a 1601 point sweep.
- Trace 1 is displaying S11 in LogMag and Phase format
- Trace 2 is displaying S12 in LogMag format
- Trace 3 is displaying S21 in Phase format
- Trace 4 is displaying S22 in Smith Chart format
- The output formatting commands CH2, FMC, and LSB are received

The number of output bytes for the O4FD query command is:

```
NBO = 12 + 8 * 4 * 1601 = 51244 bytes
```

The number of output bytes for the ORD query command is:

```
NBO = 12 + 2 * 4 * 1601 = 12820 bytes
```

The number of output bytes for the OFD3 query command is:

```
NBO = 12 + 1 * 4 * 1601 = 6416 bytes
```

The number of output bytes for the FMA or O4SR query command is:

```
NBO = 12 + 8 * 19 * 1601 = 243364 bytes
```

2-11 Input Buffer Size and NRFD Holdoff

ShockLine VNAs provide a very large input buffer that can hold up to 100 commands. Each command plus any associated data can be as large as the amount of memory available in the VNA at the time. If a programmer attempts to exceed 100 commands, the LAN will go into Not Ready For Data (NRFD) Holdoff. This Holdoff condition will hold onto the controller PC until a command is executed which frees up room for another command. Then the new command will be read in. Some controller PCs can detect this Holdoff condition and programmers interpret this condition as an Error. It is not an Error. Rather, it is a function provided by any listener device to guarantee that commands and data are not lost. It might be an ATE program error, but not a VNA error.

2-12 Synchronization of Commands

The ShockLine VNA provides synchronization by executing commands in a serial fashion. Subsequent commands will not be parsed and executed until the current SCPI command is parsed and completely executed. Indeed, as far as VNA operations are concerned, if this serial execution method is not incorporated, the VNA and the controller PC will be in chaos. Avoiding this chaos condition is so important that the IEEE488.2 standard mandated 3 commands to be provided: *OPC, *OPC?, and *WAI.

Note	For more information, see the descriptions of the *OPC, *OPC?, and *WAI commands in Chapter 3, "IEEE Commands".

This synchronization is accomplished by the SCPI parser which waits for a Completed signal from the internal interface after starting execution of the command. While the parser is waiting, it is not parsing newer commands. Subsequent commands are put into the input buffer, awaiting their turn to be parsed and executed.

2-13 Forcing the Parser to Stop Waiting

The parser will wait forever for the Completed signal as discussed above in "Synchronization of Commands". Therefore there is no SCPI command (which would itself require parsing) that can stop the parser from waiting. The controller PC will have to send a SCPI bus command called Device Clear (DCL) or Selected Device Clear (SDC). SDC is directed at a particular device address. DCL will perform the same action on all devices on the Bus. Among the required things DCL and SDC do is reset the Parser. This causes the parser to stop waiting for the 'Completed' signal and get ready to execute a command. In the **IEEE488.2 Specification**, see **IEEE488.2 Section 5.8** and **IEEE488.1 Section 4.10** for a discussion of DCL and SDC.

2-14 Aborting an RF or Hardware Calibration

From the interface, select the Abort Cal button on the ONE PORT CAL or TWO PORT CAL menus to stop the current calibration procedure. One can also send the 'ABORT' command to abort these manually initiated calibrations. However, when a calibration is initiated from the program interface, the parser is busy waiting for the 'Completed' message from the internal interface and is not available to parse the 'ABORT' command to abort the calibration. Refer to the sections above to see how that is done. Once the parser is ready for a new command, send the 'ABORT' command.

2-15 Time-Out Settings

ShockLine VNAs provide synchronization by executing commands in a serial fashion. A new command will not be parsed and executed until the previous command has finished processing. Therefore, the synchronization commands stipulated in IEEE 488.2 (`*OPC?`, `*OPC`, and `*WAI`) execute with ease.

ShockLine VNAs provide a very large input buffer that permits storing many commands until they can be executed. The ShockLine VNAs can store very large command strings (a series of characters terminated with a line feed), but only 100 command strings at most. This creates a situation where the LAN can go into Not Ready for Data (NRFD) hold off if the controller sends more than two command strings at a time. NRFD hold off will hold onto the controller until it can store the data byte that is currently being transferred. This hold off merely guarantees that no data byte will be lost from a message. Although some controllers can detect this hold off, its occurrence is of no consequence and it is how instruments are kept communicating at the same rate on the LAN.

Some commands may take a very long time to execute, such as waiting for the end of a sweep when the IF bandwidth is very low (10 Hz to 100 Hz). In spite of the long sweep time, the controller should avoid timing out because it creates a situation where synchronization and data can be lost. A controller time-out also leaves the VNA in an unknown I/O state. This unknown I/O state may not be responsive because the proper data handshake has been interrupted, thus creating a hung interface.

In many cases, the SCPI parser will also be busy participating in a long event, such as when the commands `TRS;WFS` are sent. The parser will not be available until the sweep is finished. The only way to get the parser to stop its current task and start processing new commands is to issue a command called Device Clear (DCL). This command brings the parser back to the Ready for Data (RFD) state. The command also resets the input and output buffers, which results in both input and output data being lost.

Below is a sequence that shows how to apply a time-out properly:

```
SETTIMEOUT(40000); // Sets a longer time-out for the controller.
OUTPUT ; RST // RST (reset) is a command that can take 30 seconds or longer.
ENTER ; ONP // Outputs the number of points. The actual wait occurs here.
SETTIMEOUT(20000); // Sets the time-out back to its normal value.
```

Setting the proper time-out on the controller is very important to guarantee that the SCPI will be synchronized and data will not be lost. One should choose a time-out that allows most operations to finish without problems and set the time-out to different values on those commands that require a longer time to execute. If time-outs do occur in program execution, the time-out settings for the particular commands in question should be increased. If a time-out is due to an application error, such as sending a syntax error preceding a query or an impossible state is set up such as being in HOLD and waiting for the end of the sweep, the coding error should be fixed and the time-out setting left as it is. Timing out also leaves the bus in an unknown I/O state, so a DCL should be sent to synchronize handshaking.

2-16 Trace Type Parameters and Coefficients

The following table provides a reference for the various graph types and related data types used in the ShockLine Series VNA.

Table 2-7. Trace Parameters and Coefficients (1 of 3)

Trace Name SCPI Keyword Display Trace Abbreviation	Trace Graph Format	Default Reference Level	Reference Level Range	Default Resolution	Resolution Range Parameter / Division	Default Scale Position	Scale Reference Range - Num of Vertical Div.
Group Delay **GDELay** **GDEL**	Single Rectilinear Graph	0	+/-9.9999E2	1 microsecond	1E-13 to 1E9	5	4 to 30
Imaginary **IMAGinary** **IMAG**	Single Rectilinear Graph	0	+/-9.9999E2	1 Unit (U)	1E-5 to 1E6	5	4 to 30
Linear Mag and Phase **LINPHase** **LINPH**	Double Rectilinear Graphs	Top=0 Bottom = 0	+/-9.9999E2	Top = 10 U Bottom = 45 degrees	-NA-	Top = 5 Bottom = 5	Top = 4 to 30 Bottom = 4 to 30
Log Mag and Phase **LOGPHase** **LOGPH**	Double Rectilinear Graphs	Top=0 Bottom = 0	+/-9.9999E2	Top = 10 dB Bottom = 45 degrees	Top = 1E-3 to 1E3 Bottom = 1E-2 to 1E6	Top = 5 Bottom = 5	Top = 4 to 30 Bottom = 4 to 30
Linear Mag **MLINear** **MLIN**	Single Rectilinear Graph	0	+/-9.9999E2	10 U	1E-5 to 1E6	5	4 to 30
Log Mag **MLOGarithmic** **MLOG**	Single Rectilinear Graph	0	+/-9.9999E2	10 dB	1E-3 to 1E3	5	4 to 30
Phase **PHASe** **PHAS**	Single Rectilinear Graph	0	+/-9.9999E2	45 degrees	1E-2 to 1E6	5	4 to 30
Linear Polar Lin/Phase **PLINear** **PLIN**	Polar Graph	5	1E-8 to 9.9999E2	1 U	2E-9 to 1E6	-NA-	-NA-
Linear Polar Real/Imag **PLINCOMPlex** **PLIN**	Polar Graph	5	1E-8 to 9.9999E2	1 U	2E-9 to 1E6	-NA-	-NA-

Table 2-7. Trace Parameters and Coefficients (2 of 3)

Trace Name SCPI Keyword Display Trace Abbreviation	Trace Graph Format	Default Reference Level	Reference Level Range	Default Resolution	Resolution Range Parameter / Division	Default Scale Position	Scale Reference Range - Num of Vertlcal Div.
Log Polar **Log/Phase** **PLOGarithmic** **PLOG**	Polar Graph	0	+/-9.9999E2	10 dB	1E-5 to 1E6	-NA-	-NA-
Log Polar **Real/Imag** **PLOGCOMPlex** **PLOGCOMP**	Polar Graph	0	+/-9.9999E2	10 dB	1E-5 to 1E6	-NA-	-NA-
Real **REAL** **REAL**	Single Rectilinear Graph	0	+/-9.9999E2	1 U	1E-5 to 1E6	5	4 to 30
Real and Imaginary **REIMaginary** **REIM**	Double Rectilinear Graphs	TOP = 0 Bottom = 0	+/-9.9999E2	Top = 1 U Bottom = 1 U	Top = 1E-5 to 1E6 Bottom = 1E-5 to 1E6	Top = 5 Bottom = 5	Top = 4 to 30 Bottom = 40 to 30
Smith (R + jX) **Real/Imag** **SCOMPlex** **SCOMP**	Smith Chart - Impedance (Complex)	-NA-	+/-9.9999E2	10 U	1E-5 to 1E6	-NA-	-NA-
Smith (R + jX) **Lin/Phase** **SLINear** **SLIN**	Smith Chart - Impedance (Linear)	-NA-	+/-9.9999E2	10 U	1E-5 to 1E6	-NA-	-NA-
Smith (R + jX) **Log/Phase** **SLOGarithmic** **SLOG**	Smith Chart - Impedance (Log)	-NA-	+/-9.9999E2	10 U	1E-5 to 1E6	-NA-	-NA-
Smith (R + jX) **Impedance** **SMITh** **SMIT**	Smith Chart - Impedance (Impedance)	-NA-	+/-9.9999E2	10 U	1E-5 to 1E6	-NA-	-NA-
SWR **SWR** **SWR**	Single Rectilinear Graph	0	+/-9.9999E2	10 U	1E-5 to 1E6	5	4 to 30

Table 2-7. Trace Parameters and Coefficients (3 of 3)

Trace Name SCPI Keyword Display Trace Abbreviation	Trace Graph Format	Default Reference Level	Reference Level Range	Default Resolution	Resolution Range Parameter / Division	Default Scale Position	Scale Reference Range - Num of Vertlcal Div.
Impedance Real & Imaginary **ZCOMPlex** **ZCOMP**	Double Rectilinear Graphs	Top = 0 Ohms Bottom = 0 Ohms	+/-9.9999E2	Top = 10 Ohms Bottom = 10 Ohms	Top = 1E-5 to 1E6 Bottom = 1E-5 to 1E6	Top = 5 Bottom = 5	Top = 4 to 30 Bottom = 4 to 30
Impedance Imaginary **ZIMAGinary** **ZIMAG**	Single Rectilinear Graph	0 Ohms	+/-9.9999E2	10 Ohms	1E-5 to 1E6	5	4 to 30
Impedance Magnitude **ZMAGNitude** **ZMAGN**	Single Rectilinear Graph	0 Ohms	+/-9.9999E2	10 Ohms	1E-5 to 1E6	5	4 to 30
Impedance Real **ZREAL** **ZREAL**	Single Rectilinear Graph	0 Ohms	+/-9.9999E2	10 Ohms	1E-5 to 1E6	5	4 to 30
Impedance Capacitance **ZCAPacitance** **ZCAP**	Single Rectilinear Graph	0 farads	+/-9.9999E2	1pF	1E 15 to 1E9	5	4 to 30
Impedance Inductance **ZINDuctance** **ZIND**	Single Rectilinear Graph	0 henrys	+/-9.9999E2	1 nH	1E-15 to 1E9	5	4 to 30

2-17 Input/Output Data Files

The following is a list of file types that are supported by the VNA:

Table 2-8. Supported File Types (1 of 2)

File Extension	Description	Command Compatibility
AHC	All hardware calibration file. On a per system basis, the file contains all hardware calibration data.	:MMEM:LOAD <string> | :MMEM:STORe <string>
AIC	Analog in calibration file. Per-system.	:MMEM:LOAD <string> | :MMEM:STORe <string>
ALC	ALC calibration file. Saves all available ALC calibration for all ports. Per-system.	:MMEM:LOAD <string> | :MMEM:STORe <string>
BMP	Bitmap image file of data display area.	:MMEM:STORe <string> | :MMEM:STORe:IMAGe <string>
CCF	Calibration kit coefficients file.	:MMEMory:LOAD:CKIT
CHA	Setup and Calibration file for all channels.	:MMEM:LOAD <string> | :MMEM:STORe <string>
CHX	Setup and Calibration file for a single channel.	:MMEM:LOAD <string> | :MMEM:STORe <string>
CSV	Comma separated text data file.	:MMEM:STORe <string>
EDL	Embedding/De-embedding array file.	:MMEM:LOAD <string> | :MMEM:STORe <string>
INI	Frequency Initialization and Source Initialization table files. Default name is FreqIniTable.ini (for troubleshooting only).	
JPG	JPEG image file of data display area.	:MMEM:STORE <string> :MMEM:STORe:IMAGE <string>
KIT_INFO. "Extension"	The Anritsu Calibration kit files are supported by the instrument. The files are usually bundled together on a USB memory device. When plugged into the VNA, the USB drive is identified as drive E:\. Files can be manually loaded by navigating to drive E:\ and selecting the files. The VNA then loads all of files. Alternatively, use the LKT command to load all of the calibration kit files. The calibration kit files all have a base name of "KIT_INFO.Extension" where the extension identifies the connector geometry and gender.	Command: LKT
LMT	Set up for limit lines.	:MMEMory:STORe:LIMit <string> :MMEMory:LOAD:LIMit <string>
LOG	A list of all entries in the ShockLine event log.	:MMEM:STORe <string>
MFT	Multiple Frequency Table configuration file. Default file name is FreqTable.mft (for troubleshooting).	
PNG	PNG image of data display area.	:MMEM:STORe <string> | :MMEM:STORe:IMAGe <string>
PTC	Pretune calibration file.	

Table 2-8. Supported File Types (2 of 2)

File Extension	Description	Command Compatibility
RCVR	Receiver calibration file.	:MMEM:LOAD <string> | :MMEM:STORe <string>
S1P	Data file in S1P format (see S2P below).	:MMEM:STORe <string>
S2P	Data file in S2P standard microwave simulator text format. Includes a controlled header and only one or four S-parameters are saved. If an S2P file is requested, but not all of the S-parameters are currently being measured, a value of 0 (zero) is entered for missing parameters. If a full two-port calibration is applied, all of the S-parameters are measured, even if they do not need to be displayed. The resultant S2P file is complete with all S-parameter information. S2P files can be recalled and displayed as trace memory when they are loaded into the active channel.	:MMEM:STORe <string>
SFT	Single frequency table file (for troubleshooting).	
SGS	Setup file for segmented traces.	:MMEMory:LOAD:FSEGMent <string> | :MMEMory:LOAD:ISEGMent <string>
SLC	Source Local Oscillator (Src LO) calibration file. Per-system.	:MMEM:LOAD <string> | :MMEM:STORe <string>
SQM	Source Quadrupler hardware calibration file	:MMEM:LOAD <string> | :MMEM:STORe <string>
STA	Setup file for all channels.	:MMEM:LOAD <string> | :MMEM:STORe <string>
STX	Setup file for a single channel.	:MMEM:LOAD <string> | :MMEM:STORe <string>
TDF	Active trace data memory formatted file.	:MMEMory:LOAD:MDATA <string>
TDU	Active trace data memory unformatted file.	:MMEMory:LOAD:MDATA <string>
TMZ	Ten (10) MHz calibration file. Per-system.	:MMEM:LOAD <string> | :MMEM:STORe <string>
TXT	Active channel trace data text file. Similar to the .csv format described above. A tab-delimited format with an optional descriptive heading in which the data for every trace is saved to a defined location folder. The data for each trace is saved as an X and a Y column to accommodate multiple parameters such as mixed frequency and time domain. Subsequent traces are added as additional columns.The .txt file cannot be recalled into the VNA memory.	:MMEM:STORe <string>

2-18 Status System Reporting

The VNA's status system consists of the following SCPI-defined status-reporting structures:

- The Instrument Summary Status Byte Group
- The Standard Event Status Group
- The Operation Status Group
- The Questionable Status Group

The following paragraphs describe the registers that make up a status group and explain the status information that each status group provides.

Note	Parallel Polling is not supported in the VNA.

Status Group Registers

In general, a status group consists of a condition register, a transition filter, an event register, and an enable register. Each component is briefly described in the following paragraphs.

Condition Register

The condition register is continuously updated to reflect the current status of the instrument. There is no latching or buffering for this register, it is updated in real time. Reading the contents of a condition register does not change its contents.

Transition Filter

The transition filter is a special register that specifies which types of bit state changes in the condition register will set corresponding bits in the event register.

- Negative transition filters (NTR) are used to detect condition changes from True (1) to False (0).
- Positive transition filters (PTR) are used to detect condition changes from False (0) to True (1).
- Setting both positive and negative filters True allows an event to be reported anytime the condition changes.
- Transition filters are read-write.
- Transition filters are unaffected by queries or *CLS (clear status) and *RST commands.

Event Register

The event register latches transition events from the condition register as specified by the transition filter. Bits in the event register are latched, and once set they remain set until cleared by a query or a *CLS command Event registers are read only.

Enable Register

The enable register specifies the bits in the event register that can produce a summary bit. The VNA logically ANDs corresponding bits in the event and enable registers, and ORs all the resulting bits to obtain a summary bit. Summary bits are recorded in the Summary Status Byte. Enable registers are read-write. Querying an enable register does not affect it.

Status Group Reporting

The state of certain ShockLine VNA hardware and operational events and conditions can be determined by programming the status system. Three lower status groups provide status information to the Summary Status Byte group. The Summary Status Byte group is used to determine the general nature of an event or condition and the other status groups are used to determine the specific nature of the event or condition. The following paragraphs explain the information that is provided by each status group. Programming commands for the status system, including examples of command usage, can be found in Chapter 5, "SCPI Commands: 1-Port and 2-Port VNAs"Chapter 5, "SCPI Commands: 1-Port and 2-Port VNAs".

Summary Status Byte Group

The Summary Status Byte group, consisting of the Summary Status Byte Enable register and the Summary Status Byte, is used to determine the general nature of a ShockLine VNA event or condition. The bits in the Summary Status Byte provide the following:

Table 2-9. Status Byte Group

Bit #	Bit Name	Description
0,1	Not Used	These bits are always set to 0.
2	Error Queue (ERRQ)	Set to indicate the Error Queue contains data. The Error Query command can then be used to read the error message(s) from the queue.
3	Questionable Event (QUEST)	Set to indicate the Questionable Status summary bit has been set. The Questionable Status Event register can then be read to determine the specific condition that caused the bit to be set.
4	Message Available (MAV)	Set to indicate that the VNA has data ready in its output queue.
5	Standard Event (STD)	Set to indicate that the Standard Event Status summary bit has been set. The Standard Event Status register can then be read to determine the specific event that caused the bit to be set.
6	Master Summary Status (MSS/RQS)	Set to indicate that the VNA has at least one reason to require service. This bit is also called the Master Summary Status Bit (MSS). The individual bits in the Status Byte are ANDed with their corresponding Service Request Enable Register bits, then each bit value is ORed and input to this bit.
7	Operation Event (OPER)	Set to indicate that the Operation Status summary bit has been set. The Operation Status Event register can then be read to determine the specific condition that caused the bit to be set.

Standard Event Status Group

The Standard Event Status group, consisting of the Standard Event Status register (an Event register) and the Standard Event Status Enable register, is used to determine the specific event that set bit 5 of the Summary Status Byte. The bits in the Standard Event Status register provide the following:

Table 2-10. Standard Event Status Group

Bit #	Bit Name	Description
0	Operation Complete (OP)	Set to indicate that all pending VNA operations were completed following execution of the "*OPC" command. For more information, see the descriptions of the *OPC, *OPC?, and *WAI commands in Chapter 3, "IEEE Commands".
1	Not Used	The bit is always set to 0.
2	Query Error	Set to indicate that a query error has occurred.
3	Device Dependent Error	Set to indicate that a device-dependent error has occurred.
4	Execution Error	Set to indicate that an execution error has occurred.
5	Command Error	Set to indicate that a command error (usually a syntax error) has occurred.
6	Not Used	This bit should be set to 0 (zero).
7	Power ON	Set to indicate that the VNA is powered ON and in operation.

Operation Status Group

The Operation Status group, consisting of the Operation Condition register, the Operation Positive Transition register, the Operation Negative Transition register, the Operation Event register, and the Operation Event Enable register, is used to determine the specific condition that set bit 7 in the Summary Status Byte. The bits in the Operation Event register provide the following:

Table 2-11. Operation Status Group

Bit #	Bit Name	Description
0	Calibration Complete	Set to indicate that a calibration is complete.
1	Sweep Complete	Set to indicate that a sweep is complete. Note that the Sweep Complete Bit will not be set unless the sweep was started by an appropriate trigger commands.
2-3	Not Used	These bits should be set to 0 (zero).
4	Waiting for Trigger	Set to indicate that the VNA is in an armed “wait for trigger” state.
6-15	Not Used	These bits should be set to 0 (zero).

Questionable Status Register

The Questionable Status Register consists of the Questionable Condition register, the Questionable Positive Transition register, the Questionable Negative Transition register, the Questionable Event register, and the Questionable Event Enable register.

The Questionable Status Register is used to determine the specific condition that set bit 3 in the Summary Status Byte. The bits in the Questionable Event register provide the following:

Table 2-12. Questionable Status Register

Bit #	Bit Name	Description
0	New Service Log Entry	Set to indicate that a new entry has been made to the Windows service log.
1	Limit Failure	Set to indicate that trace data is outside a limit line boundary.
2	RF Unleveled	Set to indicate that an RF unleveled condition exists.
3	Unlocked	Set to indicate that an internal PLL unlocked condition exists.
4-15	Not Used	These bits should be set to 0 (zero).

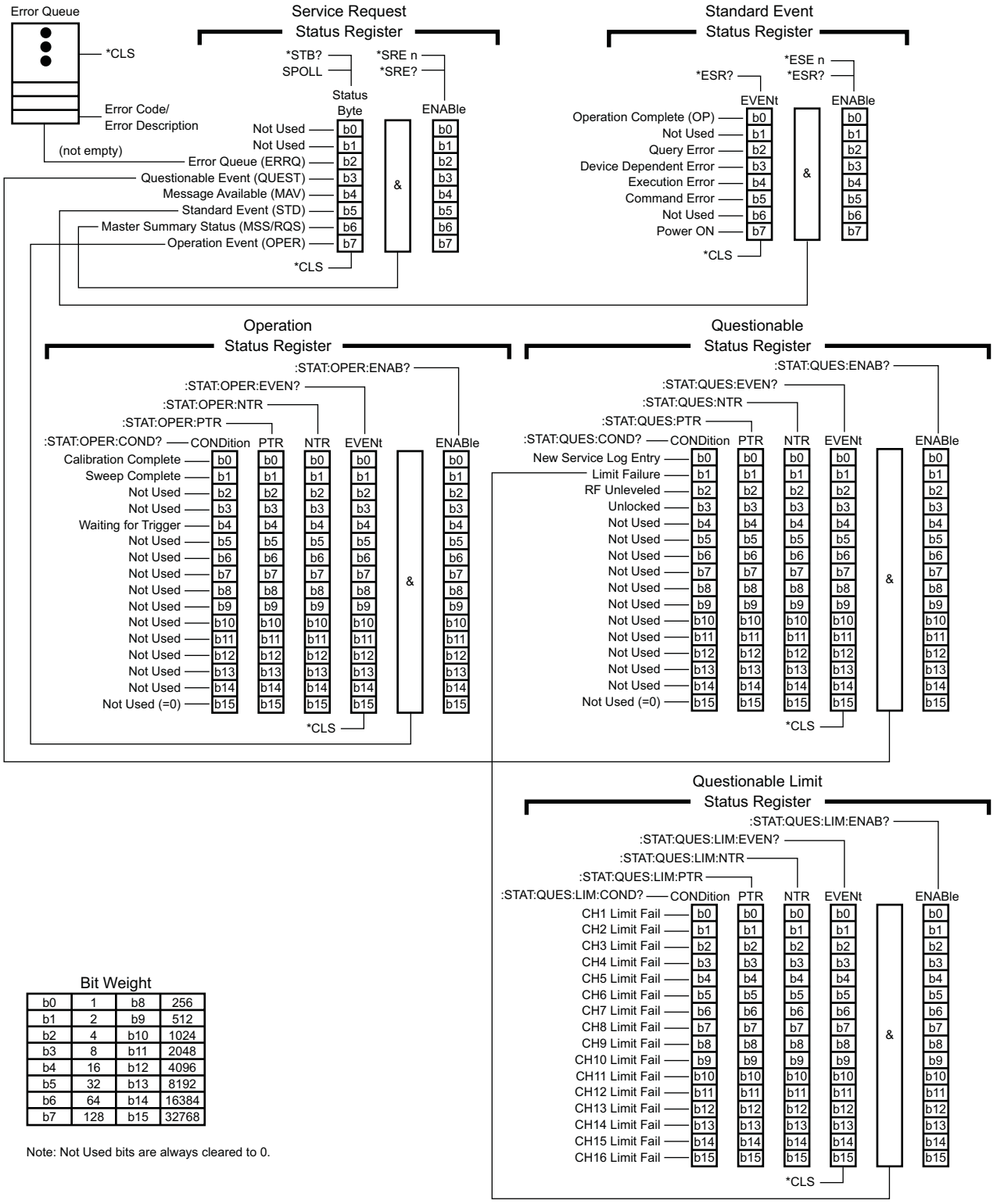
Questionable Limits Status Register

The Questionable Limits Status Register (QLSR) consists of the Questionable Limits Condition register, the Questionable Limits Event register, the Positive and Negative Transition Filters, and the Questionable Limits Event Enable register.

The QLSR is used to determine the channels that continuous limits testing failures and set Bit B1 of the Questionable Status Register. The bits in the QLSR provide the information described in the table below.

Table 2-13. Questionable Limits Status Register (QLSR)

Bit #	Bit Name	Description
0	Channel1Fail	Limits testing on Channel 1 detected a failure
1 - 15	NA	NA



Bit Weight

b0	1	b8	256
b1	2	b9	512
b2	4	b10	1024
b3	8	b11	2048
b4	16	b12	4096
b5	32	b13	8192
b6	64	b14	16384
b7	128	b15	32768

Note: Not Used bits are always cleared to 0.

Figure 2-3. Status Register Structure

2-19 Trigger System

The 2- and 4-port ShockLine VNA's trigger system is used to synchronize analyzer actions with software trigger commands. The VNA follows the layered trigger model used in SCPI instruments. The following paragraphs describe the operation of the analyzer's trigger system. A sample logic flowchart of the trigger model is shown in Figure 2-4.

Trigger Modes

The trigger system supports three trigger modes:

- **Internal Trigger Mode**

  This is an automatic triggered point-by-point measurement that is internally controlled by the DSP software.

- **External Trigger Mode**

  External mode is triggered through the rear panel input of the instrument to start a measurement based on a sweep-per-sweep trigger mode.

The following diagram is a flowchart of the triggering logic:

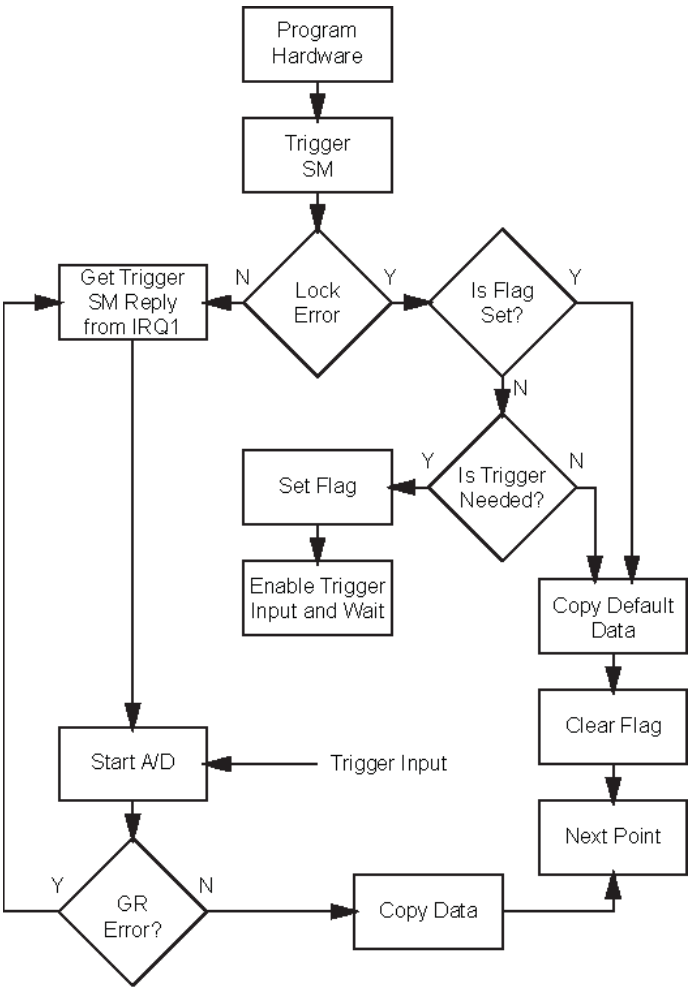


Figure 2-4. Triggering Logic Flowchart

2-20 Calibration Component Parameters

Calibration component parameter values depend on the calibration kit used and the reset status of the instrument. Table 2-14, "Loads and Through-Line Values", Table 2-15, "Default Connector Coefficients", and Table 2-16, “Connector Type Abbreviations and Descriptions” on page 2-33 summarize parameters related to calibration components, and list the factory default values for the various connector coefficients and lengths. These values may change if calibration kits are loaded that overwrite them.

A Factory Default using the :SYStem:PRESet:ZERo command restores all connector values to those in the tables below. After restoration, a Factory Default also performs a Default Default (System Default not set to USER) that changes the connector type to the appropriate connector type based on the model number.

The following general calibration component parameters should also be noted:

- A Default Default (System Default not set to USER) will change the Microstrip Kit to 10 Mil.
- A Default Default (System Default not set to USER) will change the Waveguide Kit to WR10.

Loads and Through Lines

Standard values for Loads and Through Lines are listed in the table below.

Table 2-14. Loads and Through-Line Values

Type	Parameter	Value	Units
Loads	Impedance	50	Ohms
	Resistance	50	Ohms
Through Lines	Impedance	50	Ohms
	Length	0	Meters
	Loss	0	dB
	Frequency	0	Hz

Other Connector Coefficients

Default connector coefficients apply if not overwritten by the connector values loaded from a calibration kit. Load calibration kit coefficients for the best calibration results.

Table 2-15. Default Connector Coefficients (1 of 2)

Value	SMA (male)	SMA (female)	K (male)	K (female)	N (male)	N (female)	GPC3.5 (male)	GPC3.5 (female)
OpenC0	0	0	-1e-15	-1.5e-15	65e-15	-125e-15	0	0
OpenC1	0	0	650e-27	720e-27	0	0	0	0
OpenC2	0	0	-23e-36	-23e-36	0	0	0	0
OpenC3	0	0	0.35e-45	0.35e-45	6e-45	10e-45	0	0
OpenOffsetLength	0	0	5	5	20.37	8.97	0	0
ShortL0	0	0	0	0	0	0	0	0
ShortL1	0	0	0	0	0	0	0	0
ShortL2	0	0	0	0	0	0	0	0
ShortL3	0	0	0	0	0	0	0	0

Table 2-15. Default Connector Coefficients (2 of 2)

Value	SMA (male)	SMA (female)	K (male)	K (female)	N (male)	N (female)	GPC3.5 (male)	GPC3.5 (female)
ShortOffsetLength	0	0	5	5	20.37	8.97	0	0

2-21 Calibration Command Overview

This section provides an overview of the calibration commands and when they should be used.

Setting Up a Two-Port Calibration

The commands listed in this section work on any two ports of the instrument. The FULL2 calibration is set up with the following calibration commands:

1. Calibration method

`:SENSe{1-16}:CORRection:COLLect:METHod`

Available calibration methods: LRL | LRM | SOLT | SSLT | SSST

- The LRL and LRM calibration methods are available with the "B" models only.

2. Calibration type

`:SENSe{1-16}:CORRection:COLLect:1P2PF`

`:SENSe{1-16}:CORRection:COLLect:1P2PR`

`:SENSe{1-16}:CORRection:COLLect:FULL1`

`:SENSe{1-16}:CORRection:COLLect:FULL2`

`:SENSe{1-16}:CORRection:COLLect:FULLB`

`:SENSe{1-16}:CORRection:COLLect:RESP1`

`:SENSe{1-16}:CORRection:COLLect:RESPB`

`:SENSe{1-16}:CORRection:COLLect:TFRB`

`:SENSe{1-16}:CORRection:COLLect:TFRF`

`:SENSe{1-16}:CORRection:COLLect:TFRR`

`:SENSe{1-16}:CORRection:COLLect:TYPe?`

3. Line type

`:SENSe{1-16}:CORRection:COLLect:LINE`

Available line types: COAXial | NONDISpersive

4. Load type

`:SENSe{1-16}:CORRection:COLLect:LOAD`

Available load types: FIXED | SLIDING

`:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:SELect`

Available loads: LOAD1 | LOAD2

5. Calibration port

`:SENSe{1-16}:CORRection:COLLect:PORT`

Available calibration ports: PORT1 | PORT2 | PORTP12

Defining the Calibration Standards

The following command sets the connector type:

```
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:CONNector
```

The connector types are as follows (the second letter is the gender: F for female, M for male, N for no gender):

Table 2-16. Connector Type Abbreviations and Descriptions

Connector Type	Connector Description
CF1. CM1	W1
CF2, CM2	2.4 mm
CF3, CM3	3.5 mm
CF716, CM716	7/16
CFC, CMC	TNC
CFK, CMK	K
CFKT	K (female) TOSLKF Cal Kit
CFNT	N (female) TOSLNF Cal Kit
CMKT	K (male) TOSLK Cal Kit
CMNT	N (male) TOSLN Cal Kit
CFN, CMN	N
CFN75, CMN75	N 75 ohm
CFS, CMS	SMA
CFV, CMV	V
CNG	GPC7 (no gender)
CFU1, CMU1; CFU2, CMU2; CFU3, CMU3; ...CFU32, CMU32	User Defined 1; User Defined 2; User Defined 3; ...User Defined 32

Use the following command to load a calibration kit file with its path and name as string data:

```
:MMEMory:LOAD:CKIT
```

The many calibration standard types are divided into four categories of OPEN, SHORT, LOAD, and THRU (or THROUGH) as described in the following sections.

OPEN

An OPEN standard has the following parameters that define its electrical behavior:

- C0, C1, C2 and C3 are power series coefficients used to calculate capacitance as follows:

```
C = C0 + C1*f + C2*f^2 + C3*f^3
```

These coefficients are often displayed in scientific notation as shown below:

```
C0 = number x 10E-15
C1 = number x 10E-27
C2 = number x 10E-36
C3 = number x 10E-45
```

If one enters a number for Cx whose magnitude is > 10E-5, then it is assumed that the number must be multiplied by the appropriate power of 10 shown above to determine the coefficient. Otherwise, the coefficient value is taken as is.

- OFFSET is the offset length of the load expressed in meters

Note	The parameters of the predefined types cannot be changed. Only the parameters of the User Defined types can be changed.

The following commands are used to change the OPEN standard parameters:

```
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C1
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C2
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C3
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:LABEL
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:OFFS
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:SERIAL
```

SHORT

A SHORT standard has the following parameters that define its electrical behavior:

L0, L1, L2 and L3 are power series coefficients used to calculate inductance as follows:

```
L = L0 + L1*f + L2*f^2 + L3*f^3
```

These coefficients are often displayed in scientific notation as shown below:

```
L0 = number x 10E-12
L1 = number x 10E-24
L2 = number x 10E-33
L3 = number x 10E-42
```

If one enters a number for Lx whose magnitude is > 10E-5, then it is assumed that the number must be multiplied by the appropriate power of 10 shown above to determine the coefficient. Otherwise, the coefficient value is taken as is.

- OFFSET is the offset length of the load expressed in meters

The following commands are used to change the SHORT standard parameters:

```
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L1
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L2
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L3
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:LABEL
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:OFFS
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:SERIAL
```

LOAD

A LOAD standard has the following parameters that define its electrical behavior:

- C0 is a capacitance term
- LO, L1, L2, L3 are power series coefficients used to calculate inductance as follows:

  $L = L0 + L1*f + L2*f^2 + L3*f^3$
- R is the resistance of the load
- Z0 is the characteristic impedance
- OFFSET is the offset length of the load expressed in meters

Most calibration kits have two loads; therefore, they are differentiated by naming them LOAD1 and LOAD2. Use the following commands to modify the LOAD parameters:

```
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L1
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L2
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L3
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFFS
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:R
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:Z0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:LABEL
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:SERIAL
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L1
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L2
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L3
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFFS
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:R
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:Z0
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:LABEL
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:SERIAL
```

THRU (or THROUGH)

A THRU (technically a “through”) standard has the following parameters that define its electrical behavior:

- LENGTH is the length of the line
- LOSS is the loss of the line
- FREQUENCY is the frequency at which the loss was measured
- Z0 is the Characteristic impedance
- USERECIPROCAL is not an electrical parameter. It is a flag to notify the calibrator that it should use a reciprocal type of calculation.

For 2-port VNAs, use the following commands to modify the THRU parameters:

```
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU:FREQuency
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU:LENGth
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU:LOSS
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU:Z0
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU:LABEL
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU:SERIAL
```

Performing the Calibration

After completion of the THRU parameters calibration, the actual measurements for the calibration type and methods can be performed. Each calibration type and method requires measuring the appropriate standards using the commands listed below.

2-Port measurements:

```
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD1
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD2
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD3
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD4
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD5
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD6
```

2-Port Isolation and Thru:

```
:SENSe{1-16}:CORRection:COLLect:PORT{12}:ISOL
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRU
```

Once the measurements are complete, the correction coefficients must be calculated and the calibration corrections must be applied (turned on):

```
:SENSe{1-16}:CORRection:COLLect:SAVE
:SENSe{1-16}:CORRection:ISOLation:STATe
:SENSe{1-16}:CORRection:STATe
```

To simulate a calibration, use one of the commands below to specify the calibration type:

```
:SENSe{1-16}:CORRection:COEFFicient:1P2PF
:SENSe{1-16}:CORRection:COEFFicient:1P2PR
:SENSe{1-16}:CORRection:COEFFicient:FULL1
:SENSe{1-16}:CORRection:COEFFicient:FULL2
:SENSe{1-16}:CORRection:COEFFicient:FULLB
:SENSe{1-16}:CORRection:COEFFicient:RESP1
:SENSe{1-16}:CORRection:COEFFicient:RESPB
:SENSe{1-16}:CORRection:COEFFicient:TFRB
:SENSe{1-16}:CORRection:COEFFicient:TFRF
:SENSe{1-16}:CORRection:COEFFicient:TFRR
```

The following command inputs and outputs the specified correction coefficients:

```
:SENSe{1-16}:CORRection:COEFficient
```

The correction coefficients must be specified using one of the character data arguments below:

- In 2-Port only:

  `ED1 | EP1S | ET11 | ET21 | EP2L | EX21 | ED2 | EP2S | ET22 | ET12 | EP1L | EX1`

AutoCal

The AutoCal calibration method (calibration using a compatible Precision Automatic Calibrator module) must first be specified using the command below:

```
:SENSe{1-16}:CORRection:COLLect:METHod:AUTO
```

Then the calibration type is selected using the commands described above in "Setting Up a Two-Port Calibration" on page 2-32.

The 2-port VNA calibration types supported are FULL1 and FULL2..

Note	Multiple calibrations are not supported with AutoCal.

Use the commands described above to set the desired calibration type.The following command sets the autocal box orientation manually:

```
:SENSe{1-16}:CORRection:COLLect:ECAL:ORIentation
```

This command inputs a list of up to four comma-separated items:

```
L1 | L2 | R1 | R2 | L1R2 | R1L2 | R2L1 | L2R1 |
```

The following command specifies substituting a true thru line instead of using the thru provided in the autocal box.

```
:SENSe{1-16}:CORRection:COLLect:ECAL:TRUEthru
```

The command's input argument list consists of comma separated data with alternating port selections and on/off flags. For instance, if AutoCals are being performed on Port 1 and Port 2 with true thrus on the Ports 2 then the command is:

```
:SENS1:CORR:COLL:ECAL:TRUE PORT12,ON
```

Automatic detection of the Autocal module orientation is available with the following command only with 2-Port configurations:

```
:SENSe{1-16}:CORRection:COLLect:ECAL:AUTOmatic:ORIentation[:STATe]
```

Once the AutoCal setup is complete, the following command starts the calibration:

```
:SENSe{1-16}:CORRection:COLLect:ECAL:BEGin?
```

The AutoCal calibration may require interaction with a user to perform some mechanical setup steps such as reversing the autocal box, connecting the autocal box to different port(s), or connecting external thru lines. As each step is completed, send the following command to instruct the VNA to continue with the measurements:

```
:SENSe{1-16}:CORRection:COLLect:ECAL:CONTinue?
```

The following command outputs a copy of the Autocal messages list:

:SENSe{1-16}:CORRection:COLLect:ECAL:MSGS:LIST?

This command outputs a list of up to four comma-separated items from the following list:

The following is a list of AutoCal return codes:

Table 2-17. AutoCal Module Return Code Definitions

Return Code	Code Description
0	Assurance: Assurance passed for AutoCal Modules that have an assurance step. AutoCal complete for AutoCal Modules that have no assurance step.
1	Update: AutoCal complete for AutoCal Modules that have no assurance step.
2	True Thru: Connect through line.
3	Adapter: Reverse AutoCal module connection for Adapter Removal
4	NoModule: AutoCal module not found.
5	NoOrient: AutoCal module orientation not detected.
6	NoFile: AutoCal Characterization file not found.
7	NoMatch: AutoCal Characterization file and module mismatch. Check AutoCal serial number match to AutoCal Characterization file name.
8	No12T: Characterization function needs Full 2-Port (12-Term) calibration. Full 2-Port calibration not found.
9	NotAllowed: AutoCal automatic orientation not available on Lightning modules. Orientation must be manually specified.
10	OutOfRange: Frequencies are out of AutoCal module range.
11	AssuranceFailed: Assurance failed for AutoCal modules that have an assurance step. Not applicable for AutoCal Modules that do not have an assurance step
12	Aborted: AutoCal calibration or Characterization aborted, typically by user.
13	AbortOK: Abort operation concluded successfully.
14	AbortNotOK: Abort operation not concluded successfully.
15	ACError: AutoCal unspecified error.
16	ACFatalError: AutoCal unspecified fatal error.
17	DoneCalculateCoeff: AutoCal module has completed calculating required coefficients.
18	ACConnectCalB
19	CharacBad: Characterization is bad.
20	DisplayMessage
21	ConnectToPort1: Connect AutoCal module to Port 1.
22	ConnectToPort2: Connect AutoCal module to Port 2.
25	ConnectToPorts12: Connect AutoCal module to Ports 1 and 2.
31	ConnectThrubwPorts12: Connect Thru line to Ports 1 and 2.

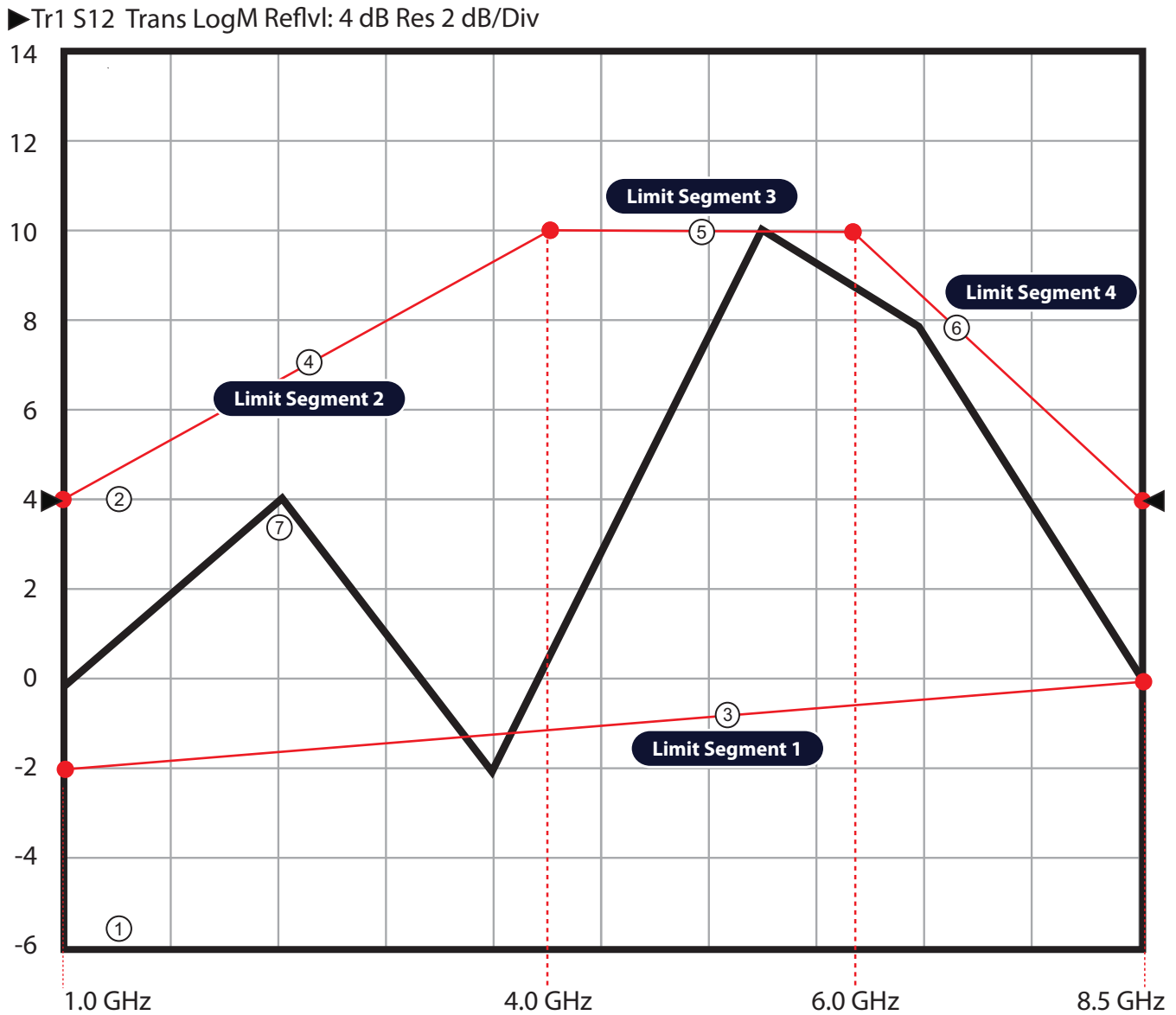
All ShockLine-compatible AutoCal modules support an assurance step.

2-22 Command Script Example – Limit Lines

This script example provides the basic procedure for establishing upper and lower limit lines for a trace.

Limit Lines for Single Rectilinear Trace Display

The requirements for this limit line example are shown in the figure below where one lower limit has been established with three upper limits.



1. Representative trace display for frequency from 1 GHz to 8.5 GHz, and levels from –6 to +14 dB.
2. Log Magnitude trace display measuring S12, with:
   - Resolution set at 2 dB per division.
   - Reference value set at 2 dB.
   - Reference position set at 5.
3. Limit Segment 1 – A lower limit line from 1 GHz at –2 dB to 8.5 GHz at 0 dB.
4. Limit Segment 2 – An upper limit line from 1 GHz at 4 dB to 4.0 GHz at 10 dB.
5. Limit Segment 3 – An upper limit line from 4.0 GHz at 10 dB to 6.0 GHz at 10 dB.
6. Limit Segment 4 – An upper limit line from 6.0 GHz at 10 dB to 8.5 GHz at 4.0 dB.
7. Typical signal trace for this DUT.

Figure 2-5. Limit Line Concept and Example

Required Equipment

ShockLine MS46322A/B Series VNA.

- ShockLine MS46322A/B Series VNA

Prerequisites

- The VNA has warmed up for at least 90 minutes.
- The calibration kit characterization file has been installed on the VNA.

DUT Requirements

The DUT measurements require the following parameters:

- S-Parameter Required: S12
- Frequency Range: 1 GHz to 8.5 GHz
- Segment 1: Lower Limit, -2 dB at 1 GHz to 0 dB at 8.5 GHz
- Segment 2: First Upper Limit, 4 dB at 1 GHz to 10 dB at 4 GHz
- Segment 3: Second Upper Limit, 10 dB at 4 GHz to 10 dB at 6 GHz
- Segment 4: Third Upper Limit, 10 dB at 6 GHz to 4 dB at 8.5 GHz

Channel and Trace Display Requirements

The following VNA setup parameters are required:

- Traces: 1
- Trace Display Type: Log Magnitude
- Trace Scale Resolution: 2 dB/Division
- Trace Scale Reference Value: 4 dB
- Trace Scale Reference Position: 5 (Positions the reference value above at the fifth gridline from the display counting from the display bottom.

VNA General Setup and Configuration

Throughout the script examples, long form commands are used for clarity. The command explanation follows the command. In this section, the VNA is cleared and per-instrument settings established. Optional commands or queries are noted and are presented for clarity.

`:SYSTem:ERRor:CLEar`

Clears the system error queue.

`:DISPlay:COLor:RESet`

Resets all colors to normal factory default value. This returns the channel, channel background, trace, limit line, and graticule colors to their default values.

`:DISPlay:COUNt 1`

Sets one (1) channel.

- If the channel display is set to a non-listed number (5, 7, 11, 13, 14, 15), the instrument is set to the next higher channel number.

`:CALCulate1:PARameter:COUNt 1`

Sets the number of traces as 1 on Channel 1.

`:DISPlay:WINDow1:SPLit R1C1`

Sets the channel display layout in a Row-by-Column format where channel window display is set to one channel on one row and one column. This is the same as maximizing a multi-channel display.

`:DISPlay:WINDow1:ACTivate 1`

Sets the active channel to the indicated channel number.

`:DISPlay:SIZe:MAXimum`

Closes the right-side menu and sets the maximum size of the graticule display.

`:CALCulate1:PARameter1:DEFine S12`

Sets the measurement parameter as S12 for Trace 1.

`:CALCulate1:PARameter1:FORMat MLOGarithmic`

Selects the display format as Log Magnitude (MLOGarithmic) for Trace 1 on Channel 1.

Frequency and Sweep Settings

In this section, the required frequency and sweep settings are established.

Examples:

`:SENSe1:FREQuency:STARt 1.0E9`

Sets channel 1 start frequency to 1 GHz.

`:SENSe1:FREQuency:STOP 8.5E9`

Sets channel 1 stop frequency to 8.5 GHz.

`:SENSe1:FREQuency:SPAN?`

Optional query. Span is automatically calculated as Stop Frequency minus Start Frequency. The query returns the resulting span in Hertz.

`7.5e9`

Frequency span is 7.5 GHz.

`:SENSe1:FREQuency:CENTer?`

Optional query. Center frequency is automatically calculated using Stop Frequency and Start Frequency as:
Fc = ((Fstop - Fstart)/2) + Fstart

`4.75 GHz`

Center frequency is `4.75` GHz

`:SENSe1:SWEep:TYPe LINear`

Sets the sweep for Channel 1 as Frequency-Based Linear.

`:SENSe1:SWEep:POINt 401`

Sets the number of measurement points for Channel 1 to 401 points.

- The minimum number of points is 2.
- The maximum number of points is limited to the total point instrument mode as 16,001.

Limit Lines Setup

There are several ways to programmatically add limit lines to a trace display. The technique below uses the :CALCulate{1-16}:SELected:LIMit command subsystem in the following general procedure:

1. Create an empty limit line segment by using the :ADD command to create a blank limit.
   - Note that each :ADD command must be followed by the :TYPE and :X1, :X2, :Y1, and :Y2 commands.
   - The first :ADD command creates a limit line that uses the entire frequency range of the instrument.
   - If the :TYPE and :X1, :X2, :Y1, and :Y2 values are not changed, no further limit line segments can be added.
2. Use the :TYPE command to define the segment as a lower or upper limit.
3. Use the :X1 and :X2 commands to configure the horizontal X-Axis start and stop points for each limit segment. In this example, the start and stop values are frequency in GHz.
4. Use the :Y1 and :Y2 commands to configure the vertical Y-Axis start and stop points for each limit segment. In this example, the start and stop values are in dB.
5. The :TYPE and :X1, :X2, :Y1, and :Y2 commands can be issued in any sequence for the segment being defined.

Clear Previous Limit Lines

Best practices recommend clearing all previous segments.

```
:CALCulate1:SELected:LIMit:SEGMent:CLEar
```

The command clears all the limit segment definitions on the active trace of the indicated channel.

Create and Configure Limit Line Segment 1

In this section, the first limit line is added, and then configured as to limit line type, start and stop frequencies, and start and stop Y-axis parameters.

Examples:

```
:CALCulate1:SELected:LIMit:SEGMent:ADD
```

On Channel 1, the command adds a limit line segment. This limit line segment will be identified as Segment 1 and set as the lower limit line across the entire frequency range of interest.

```
:CALCulate1:SELected:LIMit:SEGMent1:TYPe LOWer
```

Sets the Channel 1 Segment 1 limit line type as a lower limit line.

```
:CALCulate1:SELected:LIMit:SEGMent1:X1 1.0E9
```

Sets the Channel 1 Segment 1 lower limit line start frequency value at 1 GHz.

```
:CALCulate1:SELected:LIMit:SEGMent1:X2 8.5E9
```

Sets the Channel 1 Segment 1 lower limit line stop frequency value at 8.5 GHz.

```
:CALCulate1:SELected:LIMit:SEGMent1:Y1 -2.0
```

Sets the Channel 1 Segment 1 lower limit start Y1 value at –2.0 dB.

```
:CALCulate1:SELected:LIMit:SEGMent1:Y2 0.0
```

Sets the Channel 1 Segment 1 lower limit stop Y2 value at 0.0 dB.

Create and Configure Limit Line Segment 2

In this section, the second limit line is added, and then configured as to limit line type, start and stop frequencies, and start and stop Y-axis parameters.

```
:CALCulate1:SELected:LIMit:SEGMent:ADD
```

On Channel 1, command adds a blank limit line segment. This limit line segment will be later identified as Segment 2 and set as the first upper limit line segment.

```
:CALCulate1:SELected:LIMit:SEGMent2:TYPe UPPer
```

Sets the Channel 1 Segment 2 limit line type as an upper limit line.

```
:CALCulate1:SELected:LIMit:SEGMent2:X1 1.0E9
```

Sets the Channel 1 Segment 2 upper limit start frequency value at 1 GHz.

```
:CALCulate1:SELected:LIMit:SEGMent2:X2 4.0E9
```

Sets the Channel 1 Segment 2 upper limit line stop frequency value at 4 GHz.

`:CALCulate1:SELected:LIMit:SEGMent2:Y1 4.0`

Sets the Channel 1 Segment 2 upper limit start Y1 value at 4.0 dB.

`:CALCulate1:SELected:LIMit:SEGMent2:Y2 10.0`

Sets the Channel 1 Segment 2 upper limit stop Y2 value at 10.0 dB.

Create and Configure Limit Line Segment 3

In this section, the third limit line is added, and then configured as to limit line type, start and stop frequencies, and start and stop Y-axis parameters.

`:CALCulate1:SELected:LIMit:SEGMent:ADD`

On Channel 1, the command adds a blank limit line segment. This limit line segment will be later identified as Segment 3 and set as the second upper limit line segment.

`:CALCulate1:SELected:LIMit:SEGMent3:TYPe UPPer`

Sets the Channel 1 Segment 3 limit line type as an upper limit line.

`:CALCulate1:SELected:LIMit:SEGMent3:X1 4.0E9`

Sets the Channel 1 Segment 3 upper limit start frequency value at 4 GHz.

`:CALCulate1:SELected:LIMit:SEGMent3:X2 6.0E9`

Sets the Channel 1 Segment 3 upper limit line stop frequency value at 6 GHz.

`:CALCulate1:SELected:LIMit:SEGMent3:Y1 10.0`

Sets the Channel 1 Segment 3 upper limit start Y1 value at 10.0 dB.

`:CALCulate1:SELected:LIMit:SEGMent3:Y2 10.0`

Sets the Channel 1 Segment 3 upper limit stop Y2 value at 10.0 dB.

Create and Configure Limit Line Segment 4

In this section, the third limit line is added, and then configured as to limit line type, start and stop frequencies, and start and stop Y-axis parameters.

`:CALCulate1:SELected:LIMit:SEGMent:ADD`

On Channel 1, command adds a blank limit line segment. This limit line segment will be later identified as Segment 4 and set as the third and final upper limit line segment.

`:CALCulate1:SELected:LIMit:SEGMent4:TYPe UPPer`

Sets the Channel 1 Segment 4 limit line type as an upper limit line.

`:CALCulate1:SELected:LIMit:SEGMent4:X1 6.0E9`

Sets the Channel 1 Segment 4 upper limit start frequency value as 6 GHz.

`:CALCulate1:SELected:LIMit:SEGMent4:X2 8.5E9`

Sets the Channel 1 Segment 4 upper limit line stop frequency value at 8.5 GHz.

`:CALCulate1:SELected:LIMit:SEGMent4:Y1 10.0`

Sets the Channel 1 Segment 4 upper limit start Y1 value at 10.0 dB.

`:CALCulate1:SELected:LIMit:SEGMent4:Y2 4.0`

Sets the Channel 1 Segment 4 upper limit stop Y2 value at 4.0 dB.

`:CALCulate:LIMit:DISPlay ON`

Toggles the selected limits ON

Configure AutoCal Calibration

For this example, the Anritsu 36585K Precision Automatic Calibrator (AutoCal) Calibration Module will be used to perform the calibration. If the characterization file for the AutoCal module has not been loaded, best practices recommand using the User Interface menus to load the characterization file.

`:SENSe1:CORRection:COLLect:ECAL:AUTOmatic:ORIentation:STATe OFF`

Turn the AutoCal module automatic orientation detection off for Channel 1.

`:SENSe1:CORRection:COLLect:ECAL:ORIentation L1R2`

Set the AutoCal module orientation detection off and sets the port-to-port orientation manually for Channel 1 so that Port 1 is on the leftt and Port 2 is on the right.

`:SENSe1:CORRection:COLLect:ECAL:TRUEthru OFF`

The command turns off the use of the AutoCal True Thru feature, where a cable through is used during the AutoCal calibration for Channel 1. By setting this to OFF, the AutoCal module will use its Internal Thru capability to complete the calibration.

Ready for Measurements

The VNA is ready for measurements.

Chapter 3 — IEEE Commands

3-1 Introduction

This chapter contains all of the IEEE commands that are implemented in the instrument.

Note	When operating the ShockLine VNA through remote programming, the ShockLine application screen user interface controls are disabled. To return to local control, press the keyboard **Esc** key, or send the RTL command. For general information about GPIB, refer to Section 1-3 "IEEE 488 Description".

3-2 Command Descriptions

IEEE commands are used to control instrument status registers, status reporting, synchronization, data storage, and other common functions. All IEEE 488.2 commands are identified by the leading asterisk in the command word and are fully defined in IEEE 488.2.

Each IEEE command is followed by a complete descriptive listing of the command description, parameters, and output. If applicable, an example for each command, the default value, and the range information is written out at the end of the individual description.

- See Chapter 2, "Programming the ShockLine™ Series VNA", "Notational Conventions" on page 2-5 for definitions of parameters and notations.
- Detailed descriptions of parameter types is available in "Data Transmission Methods" on page 2-9 and through the links below.
  - <NR1>
  - <NR2>
  - <NR3>
  - <NRf>
  - <string>
  - <ASCII> or <Arbitrary ASCII>
  - <block> or <arbitrary block>
  - <char>
  - <char1>,<char2>
  - <char1>,<char2>,<char3>
  - <char1>,<char2>,<char3>,<char4>
  - MPND
  - MPNF
  - MPNI

3-3 Numeric Limits

The following numeric limits are abbreviated in the IEEE command descriptions:

- **MPND – Maximum Positive/Negative Double Precision Number**
  - +/– 1.792 693 134 86 E+308
- **MPNI – Maximum Positive/Negative Integer**
  - – 2 147 483 648 to +2 147 483 647
  - +/– 2 E31
- **MPNF – Maximum Positive/Negative Float Number**
  - +/– 3.402 819 E+38

3-4 IEEE 488.2 Commands

*CLS

Description: Clear Status Command. Clears the Status Byte, the Data Questionable Event Register, the Standard Event Status Register, the Standard Operation Status Register, the error queue, the OPC pending flag, and any other registers that are summarized in the Status Byte. No Query

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Syntax Example: `*CLS`

*DDT <Arbitrary Block> | <String>
*DDT?

Description: Define Device Trigger. The command enters the 488.2 Define Device Trigger command with an input of arbitrary block or string. The query returns the 488.2 Define Device Trigger command as an arbitrary block output string. Note that the IEEE488.2 Standard specifies the input type to only be <Arbitrary Block>. In addition, ShockLine VNAs will also accept a <String>.

Cmd Parameters: <Arbitrary Block> | <String>

Query Parameters: NA

Query Output: <Arbitrary Block>

Range: NA

Default: NA

Syntax Example: `*DDT <String>`

`*DDT?`

*ESE <NRf>
*ESE?

Description: Standard Event Status Enable and Query. The command sets the Standard Event Status Enable Register bits. The binary weighted <NRf> data parameter used with this command must have a value between 0 to 255. The query returns the value of the Standard Event Status Enable Register in <NR1> format. Refer to "Status System Reporting" on page 2-24.

Cmd Parameters: <NRf>

Query Parameters: NA

Range: 0 to 255

Query Output: <NR1>

Syntax Example: `*ESE <NRf>`

`*ESE?`

*ESR?

Description: Standard Event Status Register Query. Query only. Returns the value of the Standard Event Status Register in <NR1> format and clears the Standard Event Status Register. Refer to "Status System Reporting" on page 2-24.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <NR1>

Range: NA

Default: NA

Syntax Example: `*ESR?`

*IDN?

Description Identification Query. Query only. This query returns an instrument identification string in IEEE-488.2 specified <Arbitrary ASCII> format consisting of four fields separated by commas. The fields are: <Manufacturer>, <Model>, <Serial #>, <Firmware Revision Level> where the actual model number, serial number, and firmware version of the ShockLine™ Series VNA queried will be passed. For the MS46121A/B VNAs, the Identification Query will only display the information of the MS46121A/B using the active channel. The character output is of indeterminate length and must be the last statement issued if multiple commands and/or queries are issued at the same time. See Chapter 2 for definition of <ASCII>.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <Arbitrary ASCII>

Range: NA

Default: NA

Syntax Example: `*IDN?`

*OPC

Description: Operation Complete Command. When the *OPC command is encountered, it does nothing. Program flow is allowed to proceed to the next command in the input buffer. No query. Note that *OPC and *OPC? are not a command/query pair although they appear to be.

When the *OPC command is encountered, it immediately sets the Operational Complete bit in the Standard Events Status Register. The *OPC command is a control command for overlapped commands in systems that support command overlapping. The ShockLine does not support overlapped commands, and as a result, no pause function is provided.

Overlapped Command Background

Some non-ShockLine SCPI implementations allow commands to execute simultaneously which is called an Overlapped Command. For example, a calibration step which takes many minutes could be set into operation and then allow other communication and control commands to proceed while the calibration step continues to take place.

At some point in these non-ShockLine implementations, the separate Overlapped Operational Streams need to be brought back together and placed under programmatic control in a process called Regaining Synchronization between the controller PC and the instrument. The commands *OPC, *OPC?, and *WAI are designed to regain synchronization control and three different synchronization methods.

IEEE 488.2 Overlapped Command Definitions

The *OPC, *OPC?, and *WAI commands provide coverage for command completion before the next command is parsed and executed and comprise a class of commands termed Overlapped Commands. For instruments that support overlapped commands, each command works differently. Per the IEEE 488.2 specification:

- “IEEE 488.2 defines a distinction between overlapped and sequential commands.
- As defined in IEEE 488.2, a sequential command is one which finishes executing before the next command starts executing. An overlapped command is one which does not finish executing before the next command starts executing.
- These types of commands are described in IEEE 488.2, section 12. Examples are given in IEEE 488.2, Appendix B.
- IEEE 488.2 defines three common commands (*OPC, *OPC?, *WAI) which a device controller can use to synchronize its operation to the execution of overlapped commands.
- Each overlapped command has associated with it a Pending Operation flag.
- The device sets this flag TRUE when it passes the corresponding command from the Execution Control block to the Device Action block.
- The device sets the flag false when the device operation is finished, or has been aborted.”

*OPC Synchronization

The *OPC Operation Complete command is defined in **IEEE 488.2-1992, Section 10.18**. *OPC works only if one or more preceding commands are overlapped command. In systems that support overlapped commands, when this command is encountered, it waits until all overlapped commands have completed. Once all prior overlapped commands are complete, it instructs the parser to execute the next following command. A typical use would be to issue a *OPC before issuing a command for a long duration measurement sweep. *OPC causes the instrument to continuously sense the No Operation Pending flag. When the No Operation Pending (NOP) flag becomes TRUE, the OPC event bit in the Standard Event Status Register is set to “1” to indicate that the state of all pending operations have been completed. If this bit had been previously programmed to send a Service Request, then the controller would be aware that all is synchronized.

- For example, OV1 through OV3 are overlapped commands where the command series is OV1, OV2, OV3, *OPC, XXX.
- Commands OV1, OV2, OV3 start running. The NOP is set to FALSE.
- Parser execution stops at *OPC and the parser waits while the NOP is FALSE.
- Commands OV1, OV2, and OV3 continue to run in overlapped mode.
- Commands OV1, OV2, and OV3 are finally complete.
- The NOP flag is set to TRUE and *OPC sets the status bit to 1.
- Parser execution resumes.
- Command XXX starts running.

*OPC? Synchronization

In systems that support overlapped commands, the *OPC? Operation Complete Query command waits until all overlapped commands have been completed. When that synchronizing moment arrives, the ASCII character "1" is placed in the output buffer. This sets the MAV bit in the Status byte to TRUE which indicates there is data in the output buffer. If the controller had been attempting to read data, the moment the "1" appears in the output buffer, it is read by the controller making it aware that all is synchronized. When that synchronizing moment arrives, the next command in the input buffer is executed and the command execution is synchronized.

- For example, OV1 through OV3 are overlapped commands where the command series is OV1, OV2, OV3, *OPC?, XXX.
- Commands OV1, OV2, OV3 start running. The NOP is set to FALSE.
- Parser execution stops at *OPC? and the parser waits while the NOP is FALSE.
- Commands OV1, OV2, and OV3 continue to run in overlapped mode.
- Commands OV1, OV2, and OV3 are finally complete.
- The NOP flag is set to TRUE and *OPC? puts a one ("1") in the output buffer.
- Parser execution resumes.
- Command XXX starts running.

*WAI Synchronization

In systems that support overlapped commands, the *WAI Wait-to-Continue Command causes the parser to wait until all overlapped commands have been completed. When that synchronizing moment arrives, the next command in the input buffer is executed and command execution is synchronized.

- For example, OV1 through OV3 are overlapped commands where the command series is OV1, OV2, OV3, *WAI, XXX.
- Commands OV1, OV2, OV3 start running. The NOP is set to FALSE.
- Parser execution stops at *WAI and the parser waits while the NOP is FALSE.
- Commands OV1, OV2, and OV3 continue to run in overlapped mode.
- Commands OV1, OV2, and OV3 are finally complete.
- The NOP flag is set to TRUE.
- Parser execution resumes.
- Command XXX starts running.

ShockLine and Overlapped Commands

In ShockLine VNAs, there are no Overlapped Commands. Everything works synchronously and no command is executed until the command which preceded it has finished. Because of this, the commands *OPC, *OPC? and *WAI work slightly differently than the IEEE 488.2 specification.

Related Commands: *OPC, *OPC?, and *WAI commands.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <NR1>

Range: NA

Default: NA

Syntax Example: `*OPC`

*OPC?

Description: Operation Complete Query. Query only. Not a command/query pair with *OPC. When the *OPC? command is encountered, it does nothing. Program flow is allowed to proceed to the next command in the input buffer.

Per IEEE 488.2, *OPC? is an Overlapped Command which ShockLine VNAs do not support. When the *OPC? command is encountered, it immediately puts the character "1" (one) in the Output Queue buffer, and parser execution continues. It sets the MAV bit true when all pending operations are complete.

The *OPC, *OPC?, and *WAI Overlapped Commands are related. Because ShockLine does not support overlapped commands, they do not function exactly as specified in IEEE 488.2. See the description of Overlapped Commands in the *OPC command description above.

Related commands: *OPC, *OPC?, and *WAI commands.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <NR1>

Range: NA

Default: NA

Syntax Example: `*OPC?`

*OPT?

Description: Operation Query. Query only. The query reads out the identification number of an option installed in the ShockLine Series VNA. See Chapter 2 for definition of <ASCII>.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default: NA

Query Output: <Arbitrary ASCII>

Syntax Example: `*OPT?`

*RST

Description: Reset Command. The *RST command performs a device reset of the VNA to a pre-defined condition or to a user-defined condition. No query. The user-defined condition of *RST resets all user programmable parameters to those defined by the user in a saved configuration file. The pre-defined condition of the *RST command sets the defaults described below. For additional information on default parameter values, see each SCPI command in this manual.

*RST Does Reset the Following Parameters

Except as explicitly excluded in the next section, the *RST command does the following:

- Sets the device-specific functions to a known state that is independent of the past-use history of the device;
- Device specific commands may be provided to program a different reset state than the original factory-supplied one;
- Sets the macro defined by *DDT to a device-defined state;
- Disables macros;
- Forces the device into the OCIS state (Operation Complete Command Idle State);
- Forces the device into the OQIS state (Operation Complete Query Idle State).

*RST Does Not Reset the Following Parameters

The *RST command does not change the parameters listed below:

- Does not change the state of the IEEE 488.1 interface;
- Does not change the selected IEEE 488.1 address of the device;
- Does not change the GPIB address of the device;
- Does not change the Output Queue;
- Does not change any Event Enable Register settings including the Standard Event Status Enable Register;
- Does not change any Event Register setting including the Standard Event Status Register settings;
- Does not change the power-on-status-clear flag setting;
- Does not change the Service Request Enable Register;

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default: NA

Syntax Example: `*RST`

*SRE <NRf>
*SRE?

Description: Service Request Enable. The command sets the Service Request Enable Register bits. A zero value in the command resets the register. The query returns the value of the Service Request Enable Register in <NR1> format. Bit 6 is always zero. The integer data parameter used with this query have a value between 0 to 255.

Cmd Parameters: <NRf>

Query Parameters: NA

Query Output: <NR1>

Range: 0 to 255; 0 performs a register reset.

Default: NA

Syntax Example: `*SRE <NRf>`

`*SRE?`

*STB?

Description: Read Status Byte Query. Query only. Returns the content of the Status Byte Register (bits 0 through 5 and 7). Bit 6 is the Master Summary Status bit value. The command does not reset the status byte values.

Query Parameters: NA

Query Output: <NR1>

Range: NA

Default: NA

Syntax Example: `*STB?`

*TRG

Description: Trigger Command. Triggers the instrument if :TRIGger:SOURce{1-16} command data parameter is set to REMOTE. No query.

What is measured depends on the setting of the *DDT and :TRIGger[:SEQuence]:EXTernal:TYPe commands.

- If a sweep is selected, on 2-Port VNAs, it can be either a either a forward or reverse sweep.

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Output: NA

Syntax Example: `*TRG`

*TST?

Description: Self Test Query. Query only. Performs a self test on the active channel and outputs the self test status in <NR1> format with the following values:

- 0 indicates that self test passed.
- Any number greater than 0 indicates the number of the self test that failed.
- 144 indicates that the self test was aborted.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <NR1>

Range: NA

Default: NA

Syntax Example: `*TST?`

*WAI

Description: Wait-to-Continue Command. When the *WAI command is encountered, it does nothing. Program flow is allowed to proceed to the next command in the input buffer. No query.

The *WAI command is an Overlapped Command that provides coverage for command completion when the device supports overlapped command execution. The ShockLine does not support overlapped commands, and as a result, the *WAI does not function exactly as specified in IEEE 488.2. See the description of Overlapped Commands in the *OPC command description above.

Related commands: *OPC, *OPC?, and *WAI.

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Syntax Example: `*WAI`

Chapter 4 — Diagnostic and Troubleshooting Commands

4-1 Introduction

This chapter provides a listing and description of general system-related commands used for configuration, diagnostics, internal calibration, and troubleshooting. Complete details of each command is given in the listing following each command SCPI subsystem. If applicable, an example for each command, the default value, and the SCPI information is written out at the end of the individual description.

Note	When operating the ShockLine VNA through remote programming, the user interface controls are disabled. To return to local control, press the keyboard **Esc** key, or send the RTL command. For general information about GPIB, refer to "IEEE 488 Description" on page 1-2.

4-2 Parameters and Notations

See Chapter 2, "Notational Conventions" on page 2-5 for definitions of parameters and notations. A notation summary table is available in Table 2-4, "Parameter Notations" on page 2-6.

Detailed descriptions are available in "Data Transmission Methods" on page 2-9 and through the links below.

- <NR1>
- <NR2>
- <NR3>
- <NRf>
- <string>
- <ASCII> or <Arbitrary ASCII>
- <block> or <arbitrary block>
- <char>
- <char1>,<char2>
- <char1>,<char2>,<char3>
- <char1>,<char2>,<char3>,<char4>
- MPND
- MPNF
- MPNI

4-3 Numeric Limits

The following numeric limits are shown abbreviated in the command descriptions:

- **MPND ---** Maximum Positive/Negative Double Precision Number
  - +/– 1.792 693 134 86 E+308
- **MPNI ---** Maximum Positive/Negative Integer
  - – 2 147 483 648 to +2 147 483 647
  - +/– 2 E31
- **MPNF ---** Maximum Positive/Negative Float Number
  - +/– 3.402 819 E+38

4-4 Self Test Commands

:TST?

Description: Self Test and Output Status. Query only. The query performs an instrument self test and outputs its status.

Query Output: <NR1>

Query Parameters: NA

Command Type: System Command

:TSTRES?

Description: Self Test Results Output. Query only. The query outputs the last self test results.

Query Output: <Arbitrary Block>

Query Parameters: NA

Command Type: System Command

Chapter 5 — SCPI Commands: 1-Port and 2-Port VNAs

5-1 Introduction

This chapter contains all of the SCPI commands (required and native) that are implemented in the 1-port and 2-port instruments. Note that only those commands that are appropriate for 1-port measurements will function on the MS46121A/B.

The SCPI commands are grouped by their respective subsystems. For each subsystem, the commands are described in detail in the listing. The notation corresponds to one of the SCPI standards t9o a large extent.

Note	When operating the ShockLine VNA through remote programming, the user interface controls are disabled. To return to local control, press the keyboard **Esc** key, or send the RTL command. For general information about GPIB, refer to Section 1-3 “IEEE 488 Description”.

5-2 MS46322A/B vs. MS46122A/B, MS46121A/B SCPI Configuration

The MS46322A/B comes with an embedded computer that is pre-configured with NI VISA and will handle the TCP/IP protocol for sending SCPI commands using VXI-11. The MS46122A/B and MS46121A/B require an external PC and will send SCPI commands through a TCP/IP protocol, but the external PC configuration has some important information that is necessary for proper command handling. If the external PC has a registered version of NI VISA, the user can proceed with using VXI-11 for SCPI command support.

5-3 Minimum/Maximum Frequency Limits and Related Parameters

The minimum and maximum instrument frequencies depend on the instrument model, the installed options, and whether the VNA is a standalone unit or part of a system. See Chapter 1, “General Information”, “Minimum/Maximum Instrument Frequency and Related Parameters” on page 1-13 for additional information.

The following tables for standalone VNA frequency limits are available:

- Table 1-2, “Standalone VNAs – Default Start, Default CW, and Default Stop Frequencies” on page 1-13
- Table 1-3, “Standalone VNAs – Minimum Start, Minimum CW, and Maximum Start Frequencies” on page 1-13
- Table 1-4, “Standalone VNAs – Minimum Stop, Maximum Stop, and Maximum CW Frequencies” on page 1-14
- Table 1-5, “Standalone VNAs – Default Frequency Span and Maximum Frequency Span” on page 1-14
- Table 1-6, “Standalone VNAs – Minimum Center Frequency and Maximum Center Frequency” on page 1-15
- Table 1-7, “Standalone VNAs – Default Center Frequencies” on page 1-15

5-4 Command Level Hierarchy

The different levels of the SCPI command hierarchy are represented in a table by means of indentations to the right. Lower command levels are indented farther to the right. Observe that the complete notation of the command always includes the higher levels as well.

For example, `:SENSe{1-16}:FREQuency:CENTer` has three command levels and is represented in the table as follows:

```
:SENSe{1-16} (first level)
  :FREQuency (second level)
    :CENTer (third level)
```

The maximum number of command levels is eight. For example, the command:
`:SENSe{1-16}:CORRection:COLLect:LRL:DEVice{1-2}:PORT12:LINE:FREQuency`
has eight command levels as follows:

```
:SENSe{1-16} (first level)
  :CORRection (second level)
    :COLLect (third level)
      :LRL (fourth level)
        :DEVice{1-2} (fifth level)
          :PORT12 (sixth level)
            :LINE (seventh level)
              :FREQuency (eighth level)
```

Command Descriptions and Notation Conventions

Complete details of each command is given following the table of commands for each SCPI subsystem. If applicable, an example for each command, the default value, and the SCPI information is written out at the end of the individual description.

When a range is given in braces such as {1-16] it means supply a single value in that range.

Example: :CALCulate{12} refers to channel 12

Numeric Limits

The following numeric limits are abbreviated in the SCPI command descriptions:

- MPND – Maximum Positive/Negative Double Precision Number

  +/– 1.792 631 348 6 E+308

- MPNI – Maximum Positive/Negative Integer

  – 2 147 483 648 to +2 147 483 647

  +/– 2 E31

- MPNF – Maximum Positive/Negative Float Number

  +/– 3.402 819 E+38

Notational Conventions

See Chapter 2, “Programming the ShockLine™ Series VNA”, “Notational Conventions” on page 2-5 for definitions of parameters and notations.

Detailed descriptions of parameter types is available in “Data Transmission Methods” on page 2-9 and through the links below.

- <NR1>
- <NR2>
- <NR3>
- <NRf>
- <string>
- <ASCII> or <Arbitrary ASCII>
- <block> or <arbitrary block>
- <char>
- <char1>,<char2>
- <char1>,<char2>,<char3>
- <char1>,<char2>,<char3>,<char4>
- MPND
- MPNF
- MPNI

5-5 General Parameters

The following general parameters are defined for multiple subsystems in the sections following.

- **:CALCulate{1-16} refers to the indicated channel in the range (here, from 1 to 16).**
  If the index number is not used (the braces are empty), the command applies to channel 1.
- :DEVice{1-2} refers to the indicated device for the LRL calibration.
  If the index number is not used, the first device is used.
- :FSEGMent refers to the active frequency-based segment.
- :FSEGMent{1-50} refers to the indicated frequency-based segment.
  If the index number is not used, the command applies to the frequency-based segment 1.
- :ISEGMent refers to the active index-based segment.
- :ISEGMent{1-50} refers to the indicated index-based segment.
  If the index number is not used, the command applies index-based segment 1.
- :MARKer refers to the active marker.
- :MARKer{1-13} refers to the indicated marker where Marker 1 through 12 are standard measurement markers and Marker 13 is the reference marker. If the index number is not used, the command applies to marker 1.
- :PARameter{1-16} refers to the indicated trace.
  If the index number is not used, the command applies to trace 1.
- :PORT{1-2 | 1 | 2 | 12} refers to the indicated individual port or port pair
  If the index number is not used, the command applies to Port 1 or to the Port 1 and Port 2 pair.
- :SEGMent refers to the currently active limit line.
- :SEGMent{1-50} refers to the indicated limit line.
  If the index number is not used, the command applies to segment 1.
- :SENSe{1-16} refers to the indicated channel in the range (here, from 1 to 16).
  If the index number is not used, the command applies to channel 1.
- :THRu{12} refers to the through line between the indicated port pair.
  If the index number is not used, the command applies to the Port 1 and Port 2 pair.
- :TRACe{1-16} refers to the indicated trace.
  If the index number is not used, the command applies to trace 1.
- :WINDow{1-16} refers to the indicated channel in the range (here, from 1 to 16).
  If the index number is not used, the command applies to channel 1.

:CALCulate{1-16}:ALTernate:TRACe NAMe Subsystem

The :CALCulate{1-16}:ALTernate TRACe subsystem commands are used to assign the names to identify traces.

:CALCulate{1-16}:ALL:ALTernate:TRACe:NAMe <char>

Description: Toggles the All Alternate Trace Name select on or off.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:ALL:ALT:TRAC:NAM:ON**

:CALCulate{1-16}:ALTernate:TRACe:NAMe:STATe <char>
:CALCulate{1-16}:ALTernate:TRACe:NAMe:STATe?

Description: The command toggles the trace name on and off on the indicated channel. The query returns the trace name on/off status for the indicated channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:ALT:TRAC:NAM:STAT ON**

:CALC:ALT:TRAC:NAM:STAT?

:CALCulate{1-16}[:SELected]:ALTernate:TRACe:NAMe "user defined name"
:CALCulate{1-16}[:SELected]:ALTernate:TRACe:NAMe?

Description: The command allows entry of the trace name of the indicated channel. The query returns the trace name for the indicated channel.

Cmd Parameters: <string>

Query Parameters: N/A

Range: NA

Default Value: N/A

Syntax Example: **:CALC:ALT:TRAC:NAMe ”Sparky”**

:CALC:ALT:TRAC:NAMe?

5-6 :CALCulate{1-16}:CORRection Subsystem

The :CALCulate{1-16}:CORRection subsystem commands are used to configure and control calibrations related to adapter removal and merge calibration.

Calibration Option Subsystems

Related calibration option configuration and control subsystems are:

- ":CALCulate{1-16}:CORRection Subsystem" on page 5-5
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156

:CALCulate{1-16}:CORRection:ADAPter:REMoval

Description: Performs the adapter removal. No query.

Cmd Parameters: None

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:CORR:ADAP:REM**

:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:X <string>
:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:X?

Description: Command assigns calibration X filename to be used in adapter removal. The query outputs the calibration X filename to be used in adapter removal. The X filename refers to the calibration done with adapter connected to Port 2.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.chx' where x:\directory\filename.chx must exist.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.chx where the directory and filename must exist.

Range: NA

Default Value: NA

Syntax Example: **:CALC:CORR:ADAP:REM:CAL:X 'C:\filename.chx'**

:CALC:CORR:ADAP:REM:CAL:X?

:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:Y <string>
:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:Y?

Description: Command assigns calibration Y filename to be used in adapter removal. The query outputs the calibration Y filename to be used in adapter removal. The Y filename refers to the calibration done with adapter connected to Port 1.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.chx' where x:\directory\filename.chx must exist.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.chx

Range: NA

Default Value: NA

Syntax Example: **:CALC:CORR:ADAP:REM:CAL:Y 'C:\filename.chx'**

:CALC:CORR:ADAP:REM:CAL:Y?

:CALCulate{1-16}:CORRection:ADAPter:REMoval:LENGth <NRf>
:CALCulate{1-16}:CORRection:ADAPter:REMoval:LENGth?

Description: Command inputs the adapter length (in seconds) to be used in adapter removal. The query outputs the adapter length to be used in adapter removal.

Cmd Parameters: <NRf> The input parameter is in Seconds.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Seconds.

Range: MPND

Default Value: NA

Syntax Example: **:CALC:CORR:ADAP:REM:LENG 6.6E-11**

:CALC:CORR:ADAP:REM:LENG?

5-7 :CALCulate{1-16}:EXTRaction Subsystem

The :CALCulate{1-16}:EXTRaction subsystem commands provide configuration control and execution for network extraction functions during an instrument calibration.

Calibration Option Subsystems

Related calibration option configuration and control subsystems are:

- ":CALCulate{1-16}:CORRection Subsystem" on page 5-5
- ":CALCulate{1-16}:EXTRaction Subsystem" on page 5-7

General Parameters

The general command parameters are:

- [:METHod]:A refers to Extraction Method Type A which extracts one 2-port network using the adapter extraction method. Available on 2-Port and 4-Port VNA instruments.
- [:METHod]:B refers to Extraction Method Type B which extracts one 2-port network using a two-tier calibration. Available on 2-Port and 4-Port VNA instruments.
- [:METHod]:C refers to Extraction Method Type C which extracts two 2-port networks using inner and outer calibrations. Available on 2-Port and 4-Port VNA instruments.
- [:METHod]:D refers to Extraction Method Type D which extracts two Two-Port networks using outer calibrations only using the divide-by-two method. Available on 2-Port and 4-Port VNA instruments.

:CALCulate{1-16}:EXTRaction

Description: The command performs the network extraction after using the network extraction setup commands below. This is the same as :CALCulate{1-16}:EXTRaction[:METHod]:C described below.

No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EXTR`

:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:FILe <string>
:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:FILe?

Description: Assigns the Calibration A filename to be used in Network Extraction on the indicated channel.

Returns the Calibration A filename that is to be used in Network Extraction on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.chx' where x:\directory\filename.chx must exist.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.chx

Range: NA

Default Value: NA

Syntax Example:
```
:CALC1:EXTR:CAL:CAL 'C:\directory\cala.chx'
:CALC1:EXTR:CAL:CAL?
```

:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:PORT <char>
:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:PORT?

Description: Assigns the Calibration A port to be used in Network Extraction on the indicated channel.

Returns the Calibration A Port to be used in the Network Extraction on the indicated channel.

Cmd Parameters: <char> Filename and path in the form: 'x:\directory\filename.chx' where x:\directory\filename.chx must exist.

Query Parameters: NA

Query Output: <char> PORT1 | PORT2

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EXTR:CAL:CAL:PORT1 PORT2`

`:CALC1:EXTR:CAL:CAL:PORT?`

:CALCulate{1-16}:EXTRaction:CALibration:INNer <string>
:CALCulate{1-16}:EXTRaction:CALibration:INNer?

Description: The command assigns the inner calibration filename to be used in network extraction on the indicated channel. The query outputs the inner calibration filename to be used in network extraction on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.chx' where x:\directory\filename.chx must exist.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.chx

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EXTR:CAL:CAL:INN 'C:\filename.chx'`

`:CALC1:EXTR:CAL:CAL:INN?`

:CALCulate{1-16}:EXTRaction:CALibration:OUTer <string>
:CALCulate{1-16}:EXTRaction:CALibration:OUTer?

Description: The command assigns the outer calibration filename to be used in network extraction on the indicated channel. The query outputs the outer calibration filename to be used in network extraction on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'X:\directory\filename.chx' where x:\directory\filename.chx must exist.

Query Parameters: <char> Filename and path in the form: X:\directory\filename.chx

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EXTR:CAL:OUT 'C:\directory\filename.chx'`

`:CALC1:EXTR:CAL:OUT?`

:CALCulate{1-16}:EXTRaction:ELL1:LENGth <NRf>
:CALCulate{1-16}:EXTRaction:ELL1:LENGth?

Description: Sets the electric length 1 of the given network to be used in Network Extraction on the indicated channel. The query outputs the length 1 of the given network to be used in Network Extraction on the indicated channel.

The ELL1 value is used in the following extraction methods:

- Type A = Extract one 2-port network, with adapter extraction
- Type B = Extract one 2-port network, with two tier calibration
- Type D = Extract two 2-port networks, with outer calibration only, using divide-by-two method

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Meters.

Range: NA

Default: 0

Syntax Example:
```
:CALC1:EXTR:ELL1:LENG 2.5E-10
:CALC1:EXTR:ELL1:LENG?
```

:CALCulate{1-16}:EXTRaction:ELL2:LENGth <NRf>
:CALCulate{1-16}:EXTRaction:ELL2:LENGth?

Description: Sets the electric length 2 of the given network to be used in Network Extraction on the indicated channel. The query outputs the length 2 of the given network to be used in Network Extraction on the indicated channel.

The ELL2 value is used in the following extraction methods:

- Type A - Extract one 2-port network, with adapter extraction

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Meters.

Range: MPND

Default: 0

Syntax Example:
```
:CALC1:EXTR:ELL2:LENG 2.5E-10
:CALC1:EXTR:ELL2:LENG?
```

:CALCulate{1-16}:EXTRaction:S2P1filename:FILe <string>
:CALCulate{1-16}:EXTRaction:S2P1filename:FILe?

Description: Assigns the S2P file 1 name which receives the Extracted Network S2P data for the indicated port on the indicated channel. The query outputs the S2P file 1 name which receives the Extracted Network S2P data for the indicated port on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.s2p' where x:\directory\ must exist.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.s2p.

Range: NA

Default: NA

Syntax Example:
```
:CALC1:EXTR:S2P1:FIL 'C:\directory\filename.s2p'
:CALC1:EXTR:S2P1:FIL?
```

:CALCulate{1-16}:EXTRaction:S2P2filename:FILe <string>
:CALCulate{1-16}:EXTRaction:S2P2filename:FILe?

Description: Assigns the S2P file 2 name which receives the Extracted Network S2P data for the indicated port on the indicated channel. The query outputs the S2P file 2 name which receives the Extracted Network S2P data for the indicated port on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.s2p' where x:\directory\ must exist. See Chapter 2, "Programming the ShockLine™ Series VNA", "Notational Conventions" on page 5-2 for more information.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.s2p.

Range: NA

Default: NA

Syntax Example:
```
:CALC1:EXTR:S2P2:FIL 'C:\directory\filename.s2p'
:CALC1:EXTR:S2P2:FIL?
```

:CALCulate{1-16}:EXTRaction:SXPPortpair:PORT <char>
:CALCulate{1-16}:EXTRaction:SXPPortpair:PORT?

Description: Assigns the data port set to use when creating an S2P data file on the indicated channel. Outputs the data port set assigned to use when creating an S2P data file on the indicated channel.

Cmd Parameters: <char> PORT12

Query Parameters: NA

Query Output: <char> PORT12

Range: NA

Default: NA

Syntax Example:
```
:CALC1:EXTR:SXPP:PORT12
:CALC1:EXTR:SXPP:PORT?
```

:CALCulate{1-16}:EXTRaction:ZERo:MATCh[:STATe] <char>
:CALCulate{1-16}:EXTRaction:ZERo:MATCh[:STATe]?

Description: Sets the true/false state of the zero match terms flag on the indicated channel. Outputs the true/false state of the zero match terms flag on the given channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: NA

Query Output: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:
```
:CALC1:EXTR:ZER:MATC 1
:CALC1:EXTR:ZER:MATC?
```

:CALCulate{1-16}:EXTRaction[:METHod]:A

Description: Performs the Network Extraction using Method A on the given channel. Method Type A extracts one 2-port network using the adapter extraction method. Available on 2-Port and 4-Port VNA instruments.

No query.

The following commands must be sent before performing network extraction Method A:

- :CALCulate{1-16}:EXTRaction:CALibration:CALa:FILe
- `:CALCulate{1-16}:EXTRaction:CALibration:CALB:FILe`
- `:CALCulate{1-16}:EXTRaction:CALa:PORT`
- `:CALCulate{1-16}:EXTRaction:CALB:PORT`
- `:CALCulate{1-16}:EXTRaction:ELL1:LENGth`
- `:CALCulate{1-16}:EXTRaction:S2P1filename:FILe`

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default: NA

Syntax Example:
```
:CALC1:EXTR:METH:A
```

:CALCulate{1-16}:EXTRaction[:METHod]:B

Description: Performs the Network Extraction using Method B on the given channel. Method Type B extracts one 2-port network using a two-tier calibration. Available on 2-Port and 4-Port VNA instruments.

No query.

The following commands must be sent before performing network extraction Method B:

- :CALCulate{1-16}:EXTRaction:CALibration:CALa:FILe
- :CALCulate{1-16}:EXTRaction:CALibration:CALB:FILe
- :CALCulate{1-16}:EXTRaction:CALibration:CALa:PORT
- :CALCulate{1-16}:EXTRaction:ELL1:LENGth
- :CALCulate{1-16}:EXTRaction:S2P1filename:FILe

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default: NA

Syntax Example: :CALC1:EXTR:METH:B

:CALCulate{1-16}:EXTRaction[:METHod]:C

Description: Performs the Network Extraction using Method C on the given channel. Method Type C extracts two 2-port networks using inner and outer calibrations. Available on 2-Port and 4-Port VNA instruments.

No query.

The following commands must be sent before performing network extraction Method C:

- :CALCulate{1-16}:EXTRaction:CALibration:CALa:FILe
- :CALCulate{1-16}:EXTRaction:CALibration:CALB:FILe
- :CALCulate{1-16}:EXTRaction:SXPPortpair:PORT
- :CALCulate{1-16}:EXTRaction:S2P1filename:FILe
- :CALCulate{1-16}:EXTRaction:S2P2filename:FILe

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default: NA

Syntax Example: :CALC1:EXTR:METH:C

:CALCulate{1-16}:EXTRaction[:METHod]:D

Description: Performs the Network Extraction using Method D on the given channel. Method Type D extracts two Two-Port networks using outer calibrations only using the divide-by-two method. Available on 2-Port and 4-Port VNA instruments.

No query.

The following commands must be sent before performing network extraction Method D:

- :CALCulate{1-16}:EXTRaction:ZERO:MATCh[:STATe]
- :CALCulate{1-16}:EXTRaction:ELL1:LENGth
- :CALCulate{1-16}:EXTRaction:SXPPortpair:PORT
- :CALCulate{1-16}:EXTRaction:S2P1filename:FILe
- :CALCulate{1-16}:EXTRaction:S2P2filename:FILe

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default: NA

Syntax Example: :CALC1:EXTR:METH:D

5-8 :CALCulate{1-16}:DISPlay:MARKer Subsystem

The :CALCulate{1-16}:DISPlay:MARKer subsystem command toggles the display of all markers on and off.

Marker Subsystems

Related marker configuration, control, and reporting commands are described in the following subsystems:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:DISPlay:MARKer:ALL[:STATe] <char>
:CALCulate{1-16}:DISPlay:MARKer:ALL[:STATe]?

Description: Command turns on/off the display of all markers. Query outputs the on/off status of displaying markers. Related marker command subsystems are:

- :CALCulate{1-16}:DISPlay:MARKer:ALL[:STATe] <char>
- :CALCulate{1-16}:DISPlay:MARKer:ALL[:STATe]?

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: NA

Query Output: <char> 1 | 0

Range: NA

Default: 1

Syntax Example: **:CALC:DISP:MARK:ALL ON**

:CALC:DISP:MARK:ALL?

5-9 :CALCulate{1-16}:EOOE: Subsystem

The :CALCulate{1-16}:EOOE: subsystem commands provide configuration control, network extraction, and execution functions of the OE/EO module.

:CALCulate{1-16}:EOOE:EO4Measurment:CALCulate?

Description: De-embed the OE Device characterization data from the multiport EO calibration of the given channel and return status.

Cmd Parameters: NA

Query Parameters: <NR1> Query returns one of the following:

0 – Valid

1 – Invalid

2 – InvalidSnPFile

3 – InvalidCHXFile

4 – InvalidCalType

5 – IncompatibleFreq

6 – IncompatiblePort

7 – NoCalExist

8 – WARNING:Extrapolation

9 – InvalidPortSelection

10 – InvalidSnPFileType

11 – InvalidEOFileType

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EOOE:EO4M:CALC?`

:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile <char>
:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile?

Description: Sets the OE device characterization filename for the multiport EO measurement on the given channel. Query outputs the OE device characterization filename for the multiport EO measurement on the given channel.

Cmd Parameters: <char> Path and Filename: "c:\eofiles\myfile1.s2p"

Query Parameters: <char> Query Returns: c:\eofiles\myfile1.s2p

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EOOE:EO4M:CHAR “c:\eofiles\myfile1.s2p”`

`:CALC1:EOOE:EO4M:CHAR?`

:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile:SWAP[:STATe] <char>
:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile:SWAP[:STATe]?

Description: Sets the OE device characterization Swap flag for the multiport EO measurement on the given channel. Query outputs the OE device characterization Swap flag for the multiport EO measurement of the given channel.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: <char> 1|0

Range: NA

Default Value: 0

Syntax Example: `:CALC1:EOOE:EO4M:CHAR:SWAP ON`

`:CALC1:EOOE:EO4M:CHAR:SWAP?`

:CALCulate{1-16}:EOOE:EO4Measurment:CONFiguration <char>
:CALCulate{1-16}:EOOE:EO4Measurment:CONFiguration?

Description: Sets the configuration of the multiport EO measurement on the given channel. Query outputs the configuration of the multiport EO measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT1

Syntax Example: `:CALC1:EOOE:EO4M:CONF PORT12`

`:CALC1:EOOE:EO4M:CONF?`

:CALCulate{1-16}:EOOE:EO4Measurment:EOPort <char>
:CALCulate{1-16}:EOOE:EO4Measurment:EOPort?

Description: Sets the configuration of the multiport EO measurement on the given channel. Query outputs the configuration of the multiport EO measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT2

Syntax Example: `:CALC1:EOOE:EO4M:EOP PORT12`

`:CALC1:EOOE:EO4M:EOP?`

:CALCulate{1-16}:EOOE:EO4Measurment:OEPort <char>
:CALCulate{1-16}:EOOE:EO4Measurment:OEPort?

Description: Sets the OE port assignment for the multiport EO measurement on the given channel. Query outputs the OE port assignment for the multiport EO measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT2

Syntax Example:
```
:CALC1:EOOE:EO4M:OEP PORT12
:CALC1:EOOE:EO4M:OEP?
```

:CALCulate{1-16}:EOOE:EOMeasurment:CALCulate?

Description: De-embed the OE Device characterization data from the EO calibration of the given channel and return status. This query performs a de-embedding. In order for it to work, the user must first supply OE device characterization data with an .S2P file and EO calibration data with a .CHX file.

Cmd Parameters: NA

Query Parameters: <NR1> Query returns one of the following:

0 – Valid

1 – Invalid

2 – InvalidSnPFile

3 – InvalidCHXFile

4 – InvalidCalType

5 – IncompatibleFreq

6 – IncompatiblePort

7 – NoCalExist

8 – WARNING:Extrapolation

9 – InvalidPortSelection

10 – InvalidSnPFileType

11 – InvalidEOFileType

Range: NA

Default Value: 3

Syntax Example:
```
:CALC1:EOOE:EOM:CALC?
```

:CALCulate{1-16}:EOOE:EOMeasurment:CALFile <string>
:CALCulate{1-16}:EOOE:EOMeasurment:CALFile?

Description: Sets the calibration filename for the EO measurement on the given channel. Query outputs the calibration filename for the EO measurement of the given channel.

Cmd Parameters: <string> Path and file: "c:\eofiles\myfile1.chx"

Query Parameters: <char> Query returns: c:\eofiles\myfile1.chx

Range: N/A

Default Value: N/A

Syntax Example:
```
:CALC1:EOOE:EOM:CALF "c:\eofiles\myfile1.chx"
:CALC1:EOOE:EOM:CALF?
```

:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile <string>
:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile?

Description: Sets the OE device characterization filename for the EO measurement on the given channel. Query outputs the OE device characterization filename for the EO measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.s2p"

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default Value: N/A

Syntax Example:
```
:CALC1:EOOE:EOM:CHAR "c:\eofiles\myfile1.s2p"
:CALC1:EOOE:EOM:CHAR?
```

:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile:SWAP[:STATe] <char>
:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile:SWAP[:STATe]?

Description: Sets the OE device characterization filename for the EO measurement on the given channel. Query outputs the OE device characterization filename for the EO measurement of the given channel.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: <char> 1|0

Range: NA

Default Value: 0

Syntax Example:
```
:CALC1:EOOE:EOM:CHAR:SWAP ON
:CALC1:EOOE:EOM:CHAR:SWAP?
```

:CALCulate{1-16}:EOOE:EOMeasurment:EOPort <char>
:CALCulate{1-16}:EOOE:EOMeasurment:EOPort?

Description: Sets the EO port assignment for the EO measurement on the given channel. Query outputs the EO port assignment for the EO measurement of the given channel.

Cmd Parameters: <char> PORT1 | PORT2

Query Parameters: <char> PORT1 | PORT2

Range: NA

Default Value: PORT1

Syntax Example:
```
:CALC1:EOOE:EOM:EOP PORT1
:CALC1:EOOE:EOM:EOP?
```

:CALCulate{1-16}:EOOE:GO4Measurment:CALCulate?

Description: De-embed the OE Device characterization data from the EO calibration of the given channel and return status.

Cmd Parameters: NA

Query Parameters: <NR1> Query returns one of the following:

0 – Valid

1 – Invalid

2 – InvalidSnPFile

3 – InvalidCHXFile

4 – InvalidCalType

5 – IncompatibleFreq

6 – IncompatiblePort

7 – NoCalExist

8 – WARNING:Extrapolation

9 – InvalidPortSelection

10 – InvalidSnPFileType

11 – InvalidEOFileType

Range: Any number from the query parameters list

Default Value: 11

Syntax Example:
```
:CALC1:EOOE:GO4M:CALC?
```

:CALCulate{1-16}:EOOE:GO4Measurment:CALFile <string>
:CALCulate{1-16}:EOOE:GO4Measurment:CALFile?

Description: Sets the OE calibration filename for the multiport GO measurement on the given channel. Query outputs the OE calibration filename for the multiport GO measurement of the given channel.

Cmd Parameters: <string>"c:\eofiles\myfile1.chx"

Query Parameters: <char> c:\eofiles\myfile1.chx

Range: N/A

Default: N/A

Syntax Example: `:CALC1:EOOE:GO4M:CALF "c:\eofiles\myfile1.chx"`

`:CALC1:EOOE:GO4M:CALF?`

:CALCulate{1-16}:EOOE:GO4Measurment:CHARfile <char>
:CALCulate{1-16}:EOOE:GO4Measurment:CHARfile?

Description: Sets the OE device characterization filename for the multiport GO measurement on the given channel. Query outputs the OE device characterization filename for the multiport GO measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.s2p"

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default Value: N/A

Syntax Example: `:CALC1:EOOE:GO4M:CHAR "c:\eofiles\myfile1.s2p"`

`:CALC1:EOOE:GO4M:CHAR?`

:CALCulate{1-16}:EOOE:GO4Measurment:CONFiguration <char>
:CALCulate{1-16}:EOOE:GO4Measurment:CONFiguration?

Description: Sets the OE configuration of the multiport GO measurement on the given channel. Query outputs the OE configuration of the multiport GO measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT1

Syntax Example: `:CALC1:EOOE:GO4M:CONF PORT12`

`:CALC1:EOOE:GO4M:CONF?`

:CALCulate{1-16}:EOOE:GO4Measurment:EOPort <char>
:CALCulate{1-16}:EOOE:GO4Measurment:EOPort?

Description: Sets the EO port assignment for the multiport GO measurement on the given channel. Query outputs the EO port assignment for the multiport GO measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT1

Syntax Example:
```
:CALC1:EOOE:GO4M:EOP PORT1
:CALC1:EOOE:GO4M:EOP?
```

:CALCulate{1-16}:EOOE:GO4Measurment:OEPort <char>
:CALCulate{1-16}:EOOE:GO4Measurment:OEPort?

Description: Sets the OE port assignment for the multiport GO measurement on the given channel. Query outputs the OE port assignment for the multiport GO measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT2

Syntax Example:
```
:CALC1:EOOE:GO4M:EOP Port12
:CALC1:EOOE:GO4M:EOP?
```

:CALCulate{1-16}:EOOE:GO4Measurment:TARGetfile <string>
:CALCulate{1-16}:EOOE:GO4Measurment:TARGetfile?

Description: Sets the EO device characterization target filename to use for the multiport GO measurement on the given channel. Query outputs the EO device characterization target filename to use for the multiport GO measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.s2p"

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default Value: N/A

Syntax Example:
```
:CALC1:EOOE:GO4M:TARG "c:\eofiles\myfile1.s2p"
:CALC1:EOOE:GO4M:TARG?
```

:CALCulate{1-16}:EOOE:GOMeasurment:CALCulate?

Description: Measure and store the EO device characterization data to the target file of the given channel and return status.

Cmd Parameters: NA

Query Parameters: <NR3>

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EOOE:GOM:CALC?`

:CALCulate{1-16}:EOOE:GOMeasurment:CALFile <string>
:CALCulate{1-16}:EOOE:GOMeasurment:CALFile?

Description: Sets the calibration filename for the GO measurement on the given channel. Query outputs the calibration filename for the GO measurement of the given channel.

Cmd Parameters: <string>"c:\eofiles\myfile1.chx"

Query Parameters: <char> c:\eofiles\myfile1.chx

Range: N/A

Default: N/A

Syntax Example: `:CALC1:EOOE:GOM:CALF "c:\eofiles\myfile1.chx"`

`:CALC1:EOOE:GOM:CALF?`

:CALCulate{1-16}:EOOE:GOMeasurment:CHARfile <char>
:CALCulate{1-16}:EOOE:GOMeasurment:CHARfile?

Description: Sets the OE device characterization filename for the GO measurement on the given channel. Query outputs the OE device characterization filename for the GO measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.s2p"

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default Value: N/A

Syntax Example: `:CALC1:EOOE:GO4M:CHAR "c:\eofiles\myfile1.s2p"`

`:CALC1:EOOE:GO4M:CHAR?`

:CALCulate{1-16}:EOOE:GOMeasurment:EOPort <char>
:CALCulate{1-16}:EOOE:GOMeasurment:EOPort?

Description: Sets the EO port assignment for the GO measurement on the given channel. Query outputs the EO port assignment for the GO measurement of the given channel.

Cmd Parameters: <char> PORT1|PORT2

Query Parameters: <char> PORT1|PORT2

Range: NA

Default Value: PORT1

Syntax Example: `:CALC1:EOOE:GOM:EOP PORT1`

`:CALC1:EOOE:GOM:EOP?`

:CALCulate{1-16}:EOOE:GOMeasurment:TARGetfile <string>
:CALCulate{1-16}:EOOE:GOMeasurment:TARGetfile?

Description: Sets the EO device characterization target filename to use for the GO measurement on the given channel. Query outputs the EO device characterization target filename to use for the GO measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.s2p"

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default Value: N/A

Syntax Example: `:CALC1:EOOE:GOM:TARG "c:\eofiles\myfile1.s2p"`

`:CALC1:EOOE:GOM:TARG?`

:CALCulate{1-16}:EOOE:MSGS:LIST?

Description: Outputs a copy of the EOOE messages list.

Cmd Parameters: NA

Query Parameters: <arbitrary block data>

Range: N/A

Default Value: NA

Syntax Example: `:CALC1:EOOE:MSGS:LIST?`

```
Query returns the following list:
0 – Valid
1 – Invalid
2 – InvalidSnPFile
3 – InvalidCHXFile
4 – InvalidCalType
5 – IncompatibleFreq
6 – IncompatiblePort
7 – NoCalExist
8 – WARNING:Extrapolation
9 – InvalidPortSelection
10 – InvalidSnPFileType
11 – InvalidEOFileType
```

:CALCulate{1-16}:EOOE:OE4Measurment:CALCulate?

Description: De-embed the EO Device characterization data from the multiport OE calibration of the given channel and return status.

Cmd Parameters: NA

Query Parameters: <NR1> Query returns one of the following:

0 – Valid

1 – Invalid

2 – InvalidSnPFile

3 – InvalidCHXFile

4 – InvalidCalType

5 – IncompatibleFreq

6 – IncompatiblePort

7 – NoCalExist

8 – WARNING:Extrapolation

9 – InvalidPortSelection

10 – InvalidSnPFileType

11 – InvalidEOFileType

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EOOE:OE4M:CALC? 3`

:CALCulate{1-16}:EOOE:OE4Measurment:CALFile <string>
:CALCulate{1-16}:EOOE:OE4Measurment:CALFile?

Description: Sets the calibration filename for the multiport OE measurement on the given channel. Query outputs the calibration filename for the multiport OE measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.chx"

Query Parameters: <char> c:\eofiles\myfile1.chx

Range: N/A

Default Value: N/A

Syntax Example: `:CALC1:EOOE:OE4M:CALF "c:\eofiles\myfile1.chx"`

`:CALC1:EOOE:OE4M:CALF?`

:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile <string>
:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile?

Description: Sets the EO device characterization filename for the multiport OE measurement on the given channel. Query outputs the EO device characterization filename for the multiport OE measurement of the given channel.

Cmd Parameters: <string> “c:\eofiles\myfile1.s2p”

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default Value: N/A

Syntax Example: `:CALC1:EOOE:OE4M:CHAR “c:\eofiles\myfile1.s2p”`

`:CALC1:EOOE:OE4M:CHAR?`

:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile:SWAP[:STATe] <char>
:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile:SWAP[:STATe]?

Description: Sets the EO device characterization Swap flag for the multiport OE measurement on the given channel. Query outputs the EO device characterization Swap flag for the multiport OE measurement of the given channel.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: <char> 1|0|

Range: NA

Default Value: 0

Syntax Example:
```
:CALC1:EOOE:OE4M:CHAR:SWAP ON
:CALC1:EOOE:OE4M:CHAR:SWAP?
```

:CALCulate{1-16}:EOOE:OE4Measurment:CONFiguration <char>
:CALCulate{1-16}:EOOE:OE4Measurment:CONFiguration?

Description: Sets the configuration of the multiport OE measurement on the given channel. Query outputs the configuration of the multiport OE measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT2

Syntax Example:
```
:CALC1:EOOE:OE4M:CONF PORT12
:CALC1:EOOE:OE4M:CONF?
```

:CALCulate{1-16}:EOOE:OE4Measurment:EOPort <char>
:CALCulate{1-16}:EOOE:OE4Measurment:EOPort?

Description: Sets the EO port assignment for the multiport OE measurement on the given channel. Query outputs the EO port assignment for the multiport OE measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char> PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT1

Syntax Example:
```
:CALC1:EOOE:OE4M:EOP PORT12
:CALC1:EOOE:OE4M:EOP?
```

:CALCulate{1-16}:EOOE:OE4Measurment:OEPort <char>
:CALCulate{1-16}:EOOE:OE4Measurment:OEPort?

Description: Sets the OE port assignment for the multiport OE measurement on the given channel. Query outputs the OE port assignment for the multiport OE measurement of the given channel.

Cmd Parameters: <char> PORT12 | PORT1 | PORT2

Query Parameters: <char>PORT12 | PORT1 | PORT2

Range: NA

Default Value: PORT2

Syntax Example: `:CALC1:EOOE:OE4M:OEP PORT12`

`:CALC1:EOOE:OE4M:OEP?`

:CALCulate{1-16}:EOOE:OEMeasurment:CALCulate?

Description: De-embed the EO Device characterization data from the OE calibration of the given channel and return status.

Cmd Parameters: NA

Query Parameters: <NR1> Query returns one of the following:

0 – Valid

1 – Invalid

2 – InvalidSnPFile

3 – InvalidCHXFile

4 – InvalidCalType

5 – IncompatibleFreq

6 – IncompatiblePort

7 – NoCalExist

8 – WARNING:Extrapolation

9 – InvalidPortSelection

10 – InvalidSnPFileType

11 – InvalidEOFileType

Range: NA

Default Value: 3

Syntax Example: `:CALC1:EOOE:OEM:CALC? 3`

:CALCulate{1-16}:EOOE:OEMeasurment:CALFile<string>
:CALCulate{1-16}:EOOE:OEMeasurment:CALFile?

Description: Sets the calibration filename for the OE measurement on the given channel. Query outputs the calibration filename for the OE measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.chx"

Query Parameters: <char> c:\eofiles\myfile1.chx

Range: N/A

Default: N/A

Syntax Example:
```
:CALC1:EOOE:OEM:CALF "c:\eofiles\myfile1.chx"
:CALC1:EOOE:OEM:CALF?
```

:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile <string>
:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile?

Description: Sets the EO device characterization filename to use for the OE measurement on the given channel. Query outputs the EO device charaterization filename to use for the OE measurement of the given channel.

Cmd Parameters: <string> "c:\eofiles\myfile1.s2p"

Query Parameters: <char> c:\eofiles\myfile1.s2p

Range: N/A

Default: N/A

Syntax Example:
```
:CALC1:EOOE:OEM:CHAR "c:\eofiles\myfile1.s2p"
:CALC1:EOOE:OEM:CHAR?
```

:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile:SWAP[:STATe] <char>
:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile:SWAP[:STATe]?

Description: Sets the EO device characterization Swap flag for the OE measurement on the given channel. Query outputs the EO device characterization Swap flag for the OE measurement on the given channel.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: <char> 1|0

Range: NA

Default Value: 0

Syntax Example:
```
:CALC1:EOOE:OEM:CHAR:SWAP ON
:CALC1:EOOE:OEM:CHAR:SWAP?
```

:CALCulate{1-16}:EOOE:OEMeasurment:EOPort <char>
:CALCulate{1-16}:EOOE:OEMeasurment:EOPort?

Description: Sets the EO port assignment for the OE measurement on the given channel. Query outputs the EO port assignment for the OE measurement on the given channel.

Cmd Parameters: <char> PORT1 | PORT2

Query Parameters: <char> PORT1 | PORT2

Range: NA

Default Value: PORT1

Syntax Example: `:CALC1:EOOE:OEM:EOP PORT1`

`:CALC1:EOOE:OEM:EOP?`

:CALCulate{1-16}:EOOE:TYPE?

Description: Query only. Outputs the type of EOOE measurement that is currently loaded on the instrument.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <char> EO | OE | OO | NONE

Range: NA

Default Value: NA

Syntax Example: `:CALC1:EOOE:TYPE?`

5-10 :CALCulate{1-16}:FORMat Subsystem - SnP Data

The :CALCulate{1-16}:FORMat subsystem commands assign data ports when creating SnP data files.

I/O Configuration and File Operation Subsystems

Related subsystems for I/O configuration and file operation are:

- ":CALCulate{1-16}:FORMat Subsystem - SnP Data" on page 5-29
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":FORMat Subsystem" on page 5-122
- ":HCOPy Subsystem" on page 5-125
- ":MMEMory Subsystem" on page 5-130

:CALCulate{1-16}:FORMat:S1P:PORT <char>
:CALCulate{1-16}:FORMat:S1P:PORT?

Description: Command assigns data port to use when creating an S1P data file. The query outputs the data port assigned to use when creating an S1P data file.

Cmd Parameters: <char> PORT1 | PORT2

Query Parameters: NA

Query Output: NA

Query Output: <char> PORT1 | PORT2

Range: NA

Default: PORT1

Syntax Example: **:CALC:FORM:S1P:PORT PORT1**

:CALC:FORM:S1P:PORT?

:CALCulate{1-16}:FORMat:S2P:PORT <char>
:CALCulate{1-16}:FORMat:S2P:PORT?

Description: Assigns the data port pair to use when creating an S2P data file. Outputs the data port pair assigned to use when creating an S2P data file.

Cmd Parameters: <char> PORT12

Query Parameters: NA

Query Output: <char> PORT12

Range: NA

Default: PORT12

Syntax Example: **:CALC:FORM:S2P:PORT PORT12**

:CALC:FORM:S2P:PORT?

5-11 :CALCulate{1-16}:FSIMulator:NETWork Subsystem

The :CALCulate{1-16}:FSIMulator:NETWork subsystem commands use existing calibration files with a simulated network of various types to evaluate predicted performance. The commands apply to the active network.

Calibration Simulation Subsystems

These subsystems are used to create a calibrated state in the instrument which is followed by adding the required error correction coefficients for the required calibration type. If this approach is used, each error correction coefficient is entered by separate commands. Simulated calibration subsystems are:

- ":CALCulate{1-16}:FSIMulator:NETWork Subsystem" on page 5-30
- ":CALCulate{1-16}:FSIMulator:NETWork {1-50} Subsystem" on page 5-36
- ":SENSe{1-16}:CORRection:COEFficient:PORT" on page 5-140
- ":SENSe{1-16}:CORRection:COEFficient Subsystem" on page 5-144

:CALCulate{1-16}:FSIMulator:NETWork:ADD

Description: The command adds a blank network to be defined on the indicated channel. No query.

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC1:FSIM:NETW:ADD**

:CALCulate{1-16}:FSIMulator:NETWork:C <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:C?

Description: The command sets the current LC network capacitance value on the indicated channel. The query outputs the current LC network capacitance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Farads.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Farads.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:C 3.0E-12**

:CALC1:FSIM:NETW:C?

:CALCulate{1-16}:FSIMulator:NETWork:CLEar

Description: The command clears all networks on the indicated channel. No query.

Cmd Parameters: NA

Query Parameters: NA

Query Output: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC1:FSIM:NETW:CLE**

:CALCulate{1-16}:FSIMulator:NETWork:COUNt?

Description: Query only. The query outputs the number of embedding/de-embedding networks on the indicated channel.

Cmd Parameters: NA

Query Parameters: NA

Query Output: <NR1> The output parameter is an integer.

Range: NA

Default Value: 0

Syntax Example: **:CALC1:FSIM:NETW:COUN?**

:CALCulate{1-16}:FSIMulator:NETWork:DIELectric <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:DIELectric?

Description: The command sets the current T-Line network other dielectric value on the indicated channel. The query outputs the current T-Line network dielectric value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: NA

Query Output: <NR3> The output parameter is a unitless number.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:DIEL 2.5**

:CALC1:FSIM:NETW:DIEL?

:CALCulate{1-16}:FSIMulator:NETWork:FREQuency <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:FREQuency?

Description: The command sets the current T-Line network line loss frequency value on the indicated channel. The query outputs the current T-Line network line loss frequency value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Hertz.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:FREQ 1E7**

:CALC1:FSIM:NETW:FREQ?

:CALCulate{1-16}:FSIMulator:NETWork:L <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:L?

Description: The command sets the current LC network inductance value on the indicated channel. The query outputs the current LC network inductance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:L 3.0E-9**

:CALC1:FSIM:NETW:L?

:CALCulate{1-16}:FSIMulator:NETWork:LENGth <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:LENGth?

Description: The command sets the current T-Line network line length value on the indicated channel. The query outputs the current T-Line network line length value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:LENG 2.5E-2**

:CALC1:FSIM:NETW:LENG?

:CALCulate{1-16}:FSIMulator:NETWork:LOSS <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:LOSS?

Description: The command sets the current T-Line network line loss value on the indicated channel. The query outputs the current T-Line network line loss value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in dB/mm.

Query Parameters: NA

Query Output: <NR3> The output parameter is in dB/mm.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:LOSS 3.0E0**

:CALC1:FSIM:NETW:LOSS?

:CALCulate{1-16}:FSIMulator:NETWork:MODe <char>
:CALCulate{1-16}:FSIMulator:NETWork:MODe?

Description: The command sets the current network embed/de-embed mode on the indicated channel. The query outputs the current network embed/de-embed mode on the indicated channel.

Cmd Parameters: <char> EMBed | DEEMbed

Query Parameters: NA

Query Output: <char> EMB | DEEM

Range: NA

Default Value: EMB

Syntax Example: **:CALC1:FSIM:NETW:MOD EMB**

:CALC1:FSIM:NETW:MOD?

:CALCulate{1-16}:FSIMulator:NETWork:PORT <char>
:CALCulate{1-16}:FSIMulator:NETWork:PORT?

Description: The command sets the current network port number on the indicated channel. The query outputs the current network port number on the indicated channel.

Cmd Parameters: <char> PORT1 | PORT2 | PORT12

Query Parameters: NA

Query Output: <char> PORT1 | PORT2 | PORT12

Range: NA

Default Value: PORT1

Syntax Example: **:CALC1:FSIM:NETW:PORT PORT1**

:CALC1:FSIM:NETW:PORT?

:CALCulate{1-16}:FSIMulator:NETWork:R <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:R?

Description: The command sets the current R network resistance value on the indicated channel. The query outputs the current R network resistance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Ohms.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:R 7.5E1**

:CALC1:FSIM:NETW:R?

:CALCulate{1-16}:FSIMulator:NETWork:S2P <string>
:CALCulate{1-16}:FSIMulator:NETWork:S2P?

Description: The command sets the current network S2P filename on the indicated channel. The query outputs the current network S2P filename on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.s2p' where x:\directory\ must exist. See Chapter 2, "Programming the ShockLine™ Series VNA", "Notational Conventions" on page 2-5 for more information.

Query Parameters: NA

Query Output: <char> Filename and path in the form: x:\directory\filename.s2p.

Range: NA

Default Value: NA

Syntax Example: **:CALC1:FSIM:NETW:S2P 'C:\filename.s2p'**

:CALC1:FSIM:NETW:S2P?

:CALCulate{1-16}:FSIMulator:NETWork:SWAPs2p <char>
:CALCulate{1-16}:FSIMulator:NETWork:SWAPs2p?

Description: The command sets the current network swap S2P file data flag on the indicated channel. The query outputs the current network swap S2P file data flag on the indicated channel.

Cmd Parameters: <char> TRUE | FALSe | 1 | 0

Query Parameters: NA

Query Output: <char> 1 | 0

Range: NA

Default Value: FALS

Syntax Example: **:CALC1:FSIM:NETW:SWAP TRUE**

:CALC1:FSIM:NETW:SWAP?

:CALCulate{1-16}:FSIMulator:NETWork:TYPe <char>
:CALCulate{1-16}:FSIMulator:NETWork:TYPe?

Description: The command sets the current network type on the indicated channel. The query outputs the current network type on the indicated channel. The available network choices depend on whether the instrument is in 2-Port or 4-Port VNA mode. All 2-Port networks are available for 4-Port VNAs. The following network types are available:

Types Available for 2-Port VNA Instruments

If the instrument is in two-port mode, the following types are available:

- LS = 2-Port or 4-Port VNAs. Series inductance
- LP = 2-Port or 4-Port VNAs. Parallel inductance
- CS = 2-Port or 4-Port VNAs. Series capacitance
- CP = 2-Port or 4-Port VNAs. Parallel capacitance
- RS = 2-Port or 4-Port VNAs. Resistive series network.
- RP = 2-Port or 4-Port VNAs. Resistive parallel network.
- TLine = 2-Port or 4-Port VNAs. A defined transmission line with specifications for Impedance (Ohms), Length (Meters), Loss (dB/mm), @ Frequency (GHz), and Dielectric Value. Note that programmatically, length is entered in Meters. From the user interface, length is usually entered in millimeters.

- S2Pfile = 2-Port or 4-Port VNAs. Allows an S2P calibration file to be used.

Cmd Parameters: <char> LS | LP | CS | CP | RS | RP | TLine | S2Pfile |

Query Parameters: NA

Query Output: <char> LS | LP | CS | CP | RS | RP | TL | S2P |

Range: NA

Default Value: LSCP

Syntax Example: **:CALC1:FSIM:NETW:TYP S2PFILE**

:CALC1:FSIM:NETW:TYP?

:CALCulate{1-16}:FSIMulator:NETWork:Z0 <NRf>
:CALCulate{1-16}:FSIMulator:NETWork:Z0?

Description: The command sets the current T-Line network impedance Z0 (Z zero) value on the indicated channel. The query outputs the current T-Line network impedance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: NA

Query Output: <NR3> The output parameter is in Ohms.

Range: MPND

Default Value: 50.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW:Z0 7.5E1**

:CALC1:FSIM:NETW:Z0?

:CALCulate{1-16}:FSIMulator:NETWork[:STATe] <char>
:CALCulate{1-16}:FSIMulator:NETWork[:STATe]?

Description: The command sets the network embedding/de-embedding function on/off state on the indicated channel. The query outputs the network embedding/de-embedding function on/off state on the indicated channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC1:FSIM:NETW ON**

:CALC1:FSIM:NETW?

5-12 :CALCulate{1-16}:FSIMulator:NETWork {1-50} Subsystem

The :CALCulate{1-16}:FSIMulator:NETWork{1-50} subsystem uses existing calibration files with a simulated network of various types to evaluate predicted performance. The commands use index numbers to identify the appropriate network.

Calibration Simulation Subsystems

These subsystems are used to create a calibrated state in the instrument which is followed by adding the required error correction coefficients for the required calibration type. If this approach is used, each error correction coefficient is entered by separate commands. Simulated calibration subsystems are:

- ":CALCulate{1-16}:FSIMulator:NETWork Subsystem" on page 5-30
- ":SENSe{1-16}:CORRection:COEFficient:PORT" on page 5-140
- ":SENSe{1-16}:CORRection:COEFficient Subsystem" on page 5-144

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:C <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:C?

Description: The command modifies the indicated LC network capacitance value on the indicated channel. The query outputs the indicated LC network capacitance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Farads.

Query Parameters: <NR3> The output parameter is in Farads.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:C 3.0E-12**

:CALC1:FSIM:NETW1:C?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:DELete

Description: The command deletes the indicated network from the indicated channel. No query.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC1:FSIM:NETW1:DEL**

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:DIELectric <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:DIELectric?

Description: The command modifies the indicated T-Line network other dielectric value on the indicated channel. The query outputs the indicated T-Line network other dielectric value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:DIEL 2.5E0**

:CALC1:FSIM:NETW1:DIEL?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:FREQuency <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:FREQuency?

Description: The command modifies the indicated T-Line network line loss frequency value on the indicated channel. The query outputs the indicated T-Line network line loss frequency value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:FREQ 1.0E4**

:CALC1:FSIM:NETW1:FREQ?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:L <NRf>
:CALCulate{1-16}:FSIMulator:N

ETWork{1-50}:L?

Description: The command modifies the indicated LC network inductance value on the indicated channel. The query outputs the indicated LC network inductance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:L 5.0E-9**

:CALC1:FSIM:NETW1:L?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LENGth <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LENGth?

Description: The command modifies the indicated T-Line network line length value on the indicated channel. The query outputs the indicated T-Line network line length value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:LENG 2.5E-2**

:CALC1:FSIM:NETW1:LENG?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LOSS <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LOSS?

Description: The command modifies the indicated T-Line network line loss value on the indicated channel. The query outputs the indicated T-Line network line loss value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in dB/mm.

Query Parameters: <NR3> The output parameter is in dB/mm.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:LOSS 3.0E0**

:CALC1:FSIM:NETW1:LOSS?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:MODe <char>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:MODe?

Description: The command modifies the indicated network embed/de-embed mode on the indicated channel. The query outputs the indicated network embed/de-embed mode on the indicated channel.

Cmd Parameters: <char> EMBed | DEEMbed

Query Parameters: <char> EMB | DEEM

Range: NA

Default Value: EMB

Syntax Example: **:CALC1:FSIM:NETW1:MOD EMB**

:CALC1:FSIM:NETW1:MOD?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:PORT <char>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:PORT?

Description: The command modifies the indicated network port number on the indicated channel. The query outputs the indicated network port number on the indicated channel.

Cmd Parameters: <char> PORT1 | PORT2 | PORT12

Query Parameters: <char> PORT1 | PORT2 | PORT12 Range

NA

Default Value: PORT1

Syntax Example: **:CALC1:FSIM:NETW1:PORT PORT1**

:CALC1:FSIM:NETW1:PORT?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:R <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:R?

Description: The command modifies the indicated R network resistance value on the indicated channel. The query outputs the indicated R network resistance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:R 7.5E1**

:CALC1:FSIM:NETW1:R?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:S2P <string>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:S2P?

Description: The command modifies the indicated network S2P filename on the indicated channel. The query outputs the indicated network S2P filename on the indicated channel.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.s2p' where x:\directory\filename.s2p must exist. See Chapter 2, "Programming the ShockLine™ Series VNA", "Notational Conventions" on page 2-5 for more information.

Query Parameters: <char> Filename and path in the form: x:\directory\filename.s2p

Range: NA

Default Value: NA

Syntax Example: **:CALC1:FSIM:NETW1:S2P 'C:\filename.s2p'**

:CALC1:FSIM:NETW1:S2P?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:SWAPs2p <char>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:SWAPs2p?

Description: The command modifies the indicated network swap S2P file data flag on the indicated channel. The query outputs the indicated network swap S2P file data flag on the indicated channel.

Cmd Parameters: <char> TRUE | FALSe | 1 | 0

Query Parameters: <char> 1 | 0

Range: NA

Default Value: FALS

Syntax Example: **:CALC1:FSIM:NETW1:SWAP TRUE**

:CALC1:FSIM:NETW1:SWAP?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:TYPe <char>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:TYPe?

Description: On the indicated channel, the command modifies the indicated network type. The query outputs the indicated network type on the indicated channel. The available network choices depend on whether the instrument is in 2-Port or 4-Port VNA mode. All 2-Port networks are available for 4-Port VNAs. The following network types are available:

Types Available for 2-Port VNA Instruments

If the instrument is in two-port mode, the following types are available:

- LS = 2-Port or 4-Port VNAs. Series inductance
- LP = 2-Port or 4-Port VNAs. Parallel inductance
- CS = 2-Port or 4-Port VNAs. Series capacitance
- CP = 2-Port or 4-Port VNAs. Parallel capacitance
- RS = 2-Port or 4-Port VNAs. Resistive series network.
- RP = 2-Port or 4-Port VNAs. Resistive parallel network.
- TLine = 2-Port or 4-Port VNAs. A defined transmission line with specifications for Impedance (Ohms), Length (Meters), Loss (dB/mm), @ Frequency (GHz), and Dielectric Value. Note that programmatically, length is entered in Meters. From the user interface, length is usually entered in millimeters.
- S2Pfile = 2-Port or 4-Port VNAs. Allows an S2P calibration file to be used.

Cmd Parameters: <char> LS | LP | CS | CP | RS | RP | TLine | S2Pfile |

Query Parameters: <char> LS | LP | CS | CP | RS | RP | TL | S2P |

Range: NA

Default Value: LSCP

Syntax Example: **:CALC1:FSIM:NETW1:TYP TLine**

:CALC1:FSIM:NETW1:TYP?

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:Z0 <NRf>
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:Z0?

Description: The command modifies the indicated T-Line network impedance value on the indicated channel. The query outputs the indicated T-Line network impedance value on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in ohms.

Query Parameters: <NR3> The output parameter is in ohms.

Range: MPND

Default Value: 50.00000000000E+000

Syntax Example: **:CALC1:FSIM:NETW1:Z0 7.5E1**

:CALC1:FSIM:NETW1:Z0?

5-13 :CALCulate{1-16}:IMPedance:TRANsformation Subsystem

The :CALCulate{1-16}:IMPedance:TRANsformation subsystem commands set configuration parameters for impedance transformation.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:IMPedance:TRANsformation Subsystem" on page 5-41
- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MICrostrip Subsystem" on page 5-152
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect:WAVeguide" on page 5-199
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:R0 <NRf>
:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:R0?

Description: The command sets the resistive value of the impedance transformation on the given port and channel. If the channel is not specified, applies the resistive impedance transformation to the active port on the active channel. The query outputs the resistive value of the impedance transformation on the given port and channel.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: MPND

Default Value: 0

Syntax Example:
```
:CALC1:IMP:TRAN:PORT1:R0 25.5
:CALC1:IMP:TRAN:PORT1:R0?
```

:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:X0 <NRf>
:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:X0?

Description: Sets the reactive value of the impedance transformation on the given port and channel. The query outputs the reactive value of the impedance transformation on the given port and channel.

Cmd Parameters: <NRf> The input parameter is in Ohms

Query Parameters: <NR3> The input parameter is in Ohms.

Range: MPND

Default Value: NA

Syntax Example:
```
:CALC:IMP:TRAN:PORT:X0 1E3
:CALC:IMP:TRAN:PORT:X0?
```

:CALCulate{1-16}:IMPedance:TRANsformation[:STATe]
:CALCulate{1-16}:IMPedance:TRANsformation[:STATe]?

Description: The command toggles the impedance transformation on and off on the indicated channel. The query returns the impedance transformation on/off status for the indicated channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:IMP:TRAN ON**

:CALC:IMP:TRAN?

5-14 :CALCulate{1-16}:MARKer Subsystem

The :CALCulate{1-16}:MARKer subsystem commands provide control of the marker table display and marker coupling.

Marker Subsystems

Related marker configuration, control, and reporting commands are described in the following subsystem sections:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:MARKer:TABle[:STATe] <char>
:CALCulate{1-16}:MARKer:TABle[:STATe]?

Description: Turns the marker table display on/off. The query outputs the marker table display on/off status.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:
```
:CALC:MARK:TAB ON
:CALC:MARK:TAB?
```

:CALCulate{1-16}:MARKer:COUPle <char>
:CALCulate{1-16}:MARKer:COUPle?

Description: Toggles marker coupling on/off, which changes the active state of the markers on other traces to match that of the markers on the active trace when toggled on. For example, if Trace 2 has only Marker2 on at 2 GHz, every other trace will change to have only Marker 2 on at 2 GHz. The query outputs the marker coupling on/off state.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:
```
:CALC:MARK:COUP ON
:CALC:MARK:COUP?
```

5-15 :CALCulate{1-16}:PARameter and :PARameter{1-16}

The :CALCulate{1-16}:PARameter subsystem commands control and report on the number of traces. The :CALCulate{1-16}:PARameter{1-16} subsystem commands configure the types of parameters used and how they are displayed on each trace.

Trace Subsystems

Related trace and display subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:PARameter:COUNt <NRf>
:CALCulate{1-16}:PARameter:COUNt?

Description: Command sets number of traces. The query outputs the number of traces.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 16

Default Value: 4

Syntax Example: `:CALC:PAR:COUN 4`

`:CALC:PAR:COUN?`

:CALCulate{1-16}:PARameter{1-16}:DATA:FDATa?

Description: Command is for MS46121A/B multiple channels configuration. Inputs formatted data to display on the active channel. The instrument must be in HoldSweep. The query outputs formatted data of the active Channel.

Cmd Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Range: N/A

Default Value: N/A

Syntax Example: `:CALC1:PAR1:DATA:FDAT?`

:CALCulate{1-16}:PARameter(1-16):DATA:FMEMory?

Description: Command is for MS46121A/B multiple channels configuration. Inputs formatted data to display on the active Channel and writes it to trace memory. The command can also perform math functions. The query outputs the formatted memory of the active Channel trace.

Cmd Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Range: N/A

Default Value: N/A

Syntax Example: **:CALC1:PAR1:DATA:FMEM?**

:CALCulate{1-16}:PARAmeter(1-16):DATA:SDATa?

Description: Command is for MS46121A/B multiple channels configuration. Inputs corrected real and imaginary S-parameter data to display on the active channel's trace. The data must be real and imaginary formatted data. The command converts the active trace display appropriately. The query outputs the real/imaginary S-parameter data for the active trace.

Cmd Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Query Parameters: <Block>

Output: <Block>

Range: N/A

Default Value: N/A

Syntax Example: **:CALC1:PAR1:DATA:SDAT?**

:CALCulate{1-16}:PARAmeter(1-16):DATA:SMEMory?

Description: Command is for MS46121A/B multiple channels configuration. Inputs corrected S-parameter trace memory to display on the active channel's trace. The query outputs corrected S-parameter trace memory of the active trace.

Cmd Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> See definition of ""<block> or <arbitrary block>" on page 2-10.

Range: N/A

Default Value: N/A

Syntax Example: **:CALC1:PAR1:DATA:SMEM?**

:CALCulate{1-16}:PARameter:SELect?

Description: Command is for MS46121A/B multiple channels configuration. Inputs corrected S-parameter trace memory to display on the active channel's trace. The query outputs corrected S-parameter trace memory of the active trace.

Cmd Parameters: NA

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 16

Default Value: 1

Syntax Example: **:CALC:PAR5:SEL**

:CALC:PAR:SEL?

:CALCulate{1-16}:PARameter{1-16}:DEFine <char1> | <char1>,<char2> | <char1>,<char2>,<char3>,<char4>
:CALCulate{1-16}:PARameter{1-16}:DEFine?

Description: Sets the measurement parameter of the indicated trace. Additional command parameters (<char2>, <char3>, and <char4>) may be required depending on the selection of the first <char> parameter. If multiple command parameters are required, each additional parameter is separated by a comma. Spaces between command parameters are ignored.

Available Measurement Parameters

The first parameter <char1> defines which S-Parameter or user-defined S-Parameter is measured. The available selections are:

- S11 = No other <char> parameters are required.
- S12 = No other <char> parameters are required.
- S21 = No other <char> parameters are required.
- S22 = No other <char> parameters are required.
- USR = <char2> is also required to define the numerator,<char3> is also required to define the denominator, and <char4> is also required to define the Port number.
- MEA = Maximum Efficiency Analysis. No other <char> parameters are required.

Command Parameter Requirements when S11, S12, S21, or S22 is Selected

If the S11, S12, S21, S22 parameters are selected for <char>:

- No additional command parameters are required.
- For example, :CALC:PAR:DEF S11 sets the input-reflection coefficient measurement.
- The command syntax for S-Parameters:

  `:CALCulate{1-16}:PARameter{1-16}:DEFine <char1>`

Command Parameter Requirements when USR is Selected

If the user-defined S-Parameter of USR is selected for <char1>:

- Three additional command parameters are required as <char2>, <char3>, and <char4>.
- The second additional parameter <char2> represents the numerator. The numerator values are selected from values of A1, A2, B1, B2, or 1 (one).
- The third additional parameter <char3> represents the denominator and is also selected from values of A1, A2, B1, B2, or 1.
- The fourth additional parameter <char4> defines the power source port and is selected from PORT1, PORT2, L1, or L2.

- For example, the command :CALC:PAR:DEF USR,A1,B1,PORT1 sets a user-defined S-Parameter value of A1/B1 for Port1.
- The USR parameter is provided for users who want to deviate from standard S-Parameter measurements. A typical application of the USR parameter would be to remove external effects from antenna analysis where a test antenna is compared to a reference antenna by using a ratio such as B2/A2.
- The command syntax for user-defined S-Parameters:

  ```
  :CALCulate{1-16}:PARameter{1-16}:DEFine <char1>,<char2>,
  <char3>,<char4>
  ```

Query Outputs

The query outputs vary depending on the command parameters that were set in the command string.

Query Output for S-Parameters, Parameters

If the command sets S11, S12, S21, S22:

- The query outputs only the selected parameter.
- For example, with a command setting S11, the query returns the following:

  ```
  :CALC:PAR:DEF S11

  :CALC:PAR:DEF?

  S11
  ```

Query Output for USR

If the command sets USR for a user-defined S-Parameter, the output parameters are:

- <char1> USR
- <char2> A1 | A2 | B1 | B2 | 1
- <char3> A1 | A2 | B1 | B2 | 1
- <char4> PORT1 | PORT2
- The query outputs USR, the numerator, denominator, and the associated port. Note that the numerator and denominator are separated by a forward slash ("/").
- For example, the command and query below return the following:

  ```
  :CALC:PAR:DEF USR,A1,B1,PORT1

  :CALC:PAR:DEF?

  USR,A1/B1,PORT1
  ```

Cmd Parameters: The required command parameters depend on the first parameter selected:

- <char1> S11 | S12 | S21 | S22 | USR | MEA
- <char2> A1 | A2 | A3 | B1 | B2 | 1
- <char3> A1 | A2 | A3 | B1 | B2 | 1
- <char4> PORT1 | PORT2 | L1 | L2

Query Parameters: The query parameters returned depend on which S-Parameter was selected for the command:

- <char1> S11 | S12 | S21 | S22 | USR | MEA
- <char2> A1 | A2 | B1 | B2 | 1
- <char3> A1 | A2 | B1 | B2 | 1

- <char4> PORT1 | PORT2 | L1 | L2

Output: <char1> | <char1>,<char2> | <char1>,<char2>/<char3>,<char4>

Range: NA

Default Value: The command and query default values depend on the which parameter was selected for <char1>:

- S11 for PAR
- S12 for PAR2
- S21 for PAR3
- S22 for PAR4
- S11 for all others

Syntax Example: `:CALC:PAR:DEF S11`

`:CALC:PAR:DEF USR A1,B1,PORT1`

`:CALC:PAR:DEF?`

:CALCulate{1-16}:PARameter{1-16}:FORMat <char>
:CALCulate{1-16}:PARameter{1-16}:FORMat?

Description: Selects the display format of the indicated trace. See Chapter 2, "Programming the ShockLine™ Series VNA" for more information. The available display types are:

- ETAmax = η Max (MEA-only)
- GDELay = Group Delay
- IMAGinary = Imaginary
- KQ = kQ quality (MEA-only)
- KQETAmax = kQ quality and η Max (MEA-only)
- LINPHase = Linear Mag and Phase
- LOGPHase = Log Magnitude and Phase
- MLINear = Linear Magnitude
- MLOGarithmic = Log Magnitude
- PHASe = Phase
- PLINear = Linear Polar and Linear Phase
- PLINCOMPlex = Linear Polar and Real/Imaginary
- PLOGarithmic = Log Polar and Log Phase
- PLOGCOMPlex = Log Polar and Real/Imaginary
- REAL = Real
- REIMaginary = Real and Imaginary
- SCOMPlex = Smith (R + jX) Impedance Real/Imaginary
- SLINear = Smith (R + jX) Impedance Linear/Phase
- SLOGarithmic = Smith (R + jX) Impedance Log/Phase
- SMITh = Smith (R + jX) Impedance
- SWR = SWR
- ZREAL = Impedance Real
- ZCAPacitance = Impedance Capacitance
- ZCOMPlex = Impedance Real and Imaginary
- ZIMAGinary = Impedance Imaginary
- ZINDuctance = Impedance Inductance
- ZMAGNitude = Impedance Magnitude

The query outputs the display format of the indicated trace. See Chapter 2, "Programming the ShockLine™ Series VNA" for more information and a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <char> ETAmax | GDELay | IMAGinary | KQ | KQETAmax | LINPHase | LOGPHase | MLINear | MLOGarithmic | PHASe | PLINear | PLINCOMPlex | PLOGarithmic | PLOGCOMPlex | REAL | REIMaginary | SCOMPlex | SLINear | SLOGarithmic | SMITh | SWR | ZREAL |ZCAPacitance | ZCOMPlex | ZIMAGinary | ZINDuctance | ZMAGNitude

Query Parameters: <char> ETA | GDEL | IMAG | KQ | KQETA | LINPH | LOGPH | MLIN | MLOG | PHAS | PLIN | PLINCOMP | PLOG | PLOGCOMP | REAL | REIM | SCOMP | SLIN | SLOG | SMIT | SWR | ZREAL |ZCAP | ZCOMP | ZIMAG |ZIND | ZMAGN

Range: NA

Default Value: The command/query default values depend on the value of the PARameter keyword as it varies from PAR to PAR16 listed below:

- MLOG for PAR1 to PAR16

Output: <char>

Syntax Example: **:CALC:PAR:FORM MLOG**

:CALC:PAR:FORM?

:CALCulate{1-16}:PARameter{1-16}:IMPedance:LC[:STATe] <char>
:CALCulate{1-16}:PARameter{1-16}:IMPedance:LC[:STATe]?

Description: Toggles the display of marker LC value on impedance measurements of the indicated trace on/off.

Effects only apply to PLINear, PLINCOMPlex, PLOGarithmic, PLOGCOMPlex ,and SMITH graph types.

The query returns the status of the marker LC value display on the indicated trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: :CALC:PAR1:IMP:LC ON

:CALC:PAR1:IMP:LC?

:CALCulate{1-16}:PARameter{1-16}:MEA:DEFine <NRf>
:CALCulate{1-16}:PARameter{1-16}:MEA:DEFine?

Description: Changes the ports measured for the indicated trace's MEA measurement.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 12

Default Value: 12

Syntax Example: :CALC:PAR1:MEA:DEF 12

:CALC:PAR1:MEA:DEF?

5-16 :CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem

The :CALCulate{1-16}:PARameter{1-16}:MARKer subsystem commands control the active marker, the discrete marker mode, and whether the markers are displayed.

Marker Subsystems

Related marker configuration, control, and reporting commands are described in the following subsystems:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:PARameter{1-16}:MARKer:ACTivate?

Description: Query only. The query outputs the number of the active marker on the indicated trace. If there is no active marker, the query returns a 0 (zero). Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

Note	The instrument user interface supports six measurement markers and 1 reference marker. SCPI supports up to 13 markers. The user interface reference marker is equivalent to SCPI marker 13.

Cmd Parameters: NA

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 13

Default Value: 0

Syntax Example: **:CALC:PAR5:MARK1:ACT**

:CALC:PAR5:MARK:ACT?

:CALCulate{1-16}:PARameter{1-16}:MARKer:DISCrete <char>
:CALCulate{1-16}:PARameter{1-16}:MARKer:DISCrete?

Description: Turns the discrete marker mode on/off on the given trace. The query outputs the discrete marker mode on/off state on the given trace.

- Discrete ON: Marker will only lock onto measured trace points
- Discrete OFF: Markers can be placed between measured points

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 1

Syntax Example: **:CALC:PAR:MARK:DISC ON**

:CALC:PAR:MARK:DISC?

:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:ACTivate

Description: Makes the indicated marker the active marker on the indicated trace. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker. No query.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:PAR:MARK1:ACT**

:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:X <NRf>
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:X?

Description: Enters the frequency, distance, or time value of the indicated marker and turns the marker on. The example below supposes that a point exists at 10.0E6 and is within the predefined frequency range. The query outputs the frequency, distance, or time value of the indicated marker. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

Cmd Parameters: <NRf> The input parameter is in Hertz, Meters, or Seconds.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: The range is user-defined by a start and stop frequency, a start and stop distance, or a start and stop time.

Default Value: For the X-axis, the default is the Minimum Instrument Frequency.

Syntax Example: **:CALC:PAR:MARK1:X 10.0E6**

:CALC:PAR:MARK1:X?

:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:Y?

Description: Query only. The query outputs the response value of the indicated marker. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker. Use the following command to set the display type:

```
:CALCulate{1-16}:PARameter{1-16}:FORMat
```

See Chapter 2, "Programming the ShockLine™ Series VNA" for more information and a complete listing of trace graph types, default settings, and available ranges.

Query Parameters: <NR3> | <NR3>,<NR3> The output parameter depends on the display type.

Range: NA

Default Value: NA

Syntax Example: **:CALC:PAR:MARK1:Y?**

:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}[:STATe] <char>
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}[:STATe]?

Description: For the indicated trace, the command toggles the indicated marker display on/off. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

For the indicated the indicated trace, the query outputs the marker display on/off status of the indicated marker. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:PAR:MARK1 ON**

:CALC:PAR:MARK1?

5-17 :CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem

The :CALCulate{1-16}:PARameter{1-16}:MSTatistics subsystem commands control the marker data table display and output its values.

Marker Subsystems

Related marker configuration, control, and reporting commands are described in the following subsystems:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:PARameter{1-16}:MSTatistics <char>
:CALCulate{1-16}:PARameter{1-16}:MSTatistics?

Description: For the indicated trace, the command toggles the marker statistics display on and off. Markers do not need to be turned on to display the statistics data. If the statistics are toggled on, the statistics are determined for the entire sweep range. The three values displayed are Mean (Mean), Standard Deviation (Std Dev), and Peak-to-Peak (P-P), each measured in dB. Markers do not need to be displayed to output the statistics. If the marker statistics are displayed, a label of Stat Range Entire Sweep (Statistics Range - Entire Sweep) appears above the statistic values. The query, for the indicated trace, outputs the marker statistics display on/off state.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:PAR:MST ON**

:CALC:PAR:MST?

:CALCulate{1-16}:PARameter{1-16}:MSTatistics:DATA?

Description: Query only. For the indicated trace, the query outputs the three marker statistics of the upper display. The statistics are determined for the entire sweep range. The three values output are Mean (Mean), Standard Deviation (Std Dev), and Peak-to-Peak (P-P), each measured in dB. The values are measured for the entire sweep range for the indicated trace. Markers do not need to be on to output (or display) the statistics.

Cmd Parameters: NA

Query Parameters: <NR3>, <NR3>, <NR3> Outputs marker on-screen statistics in units of dB.

Range: NA

Default Value: NA

Syntax Example: **:CALC:PAR:MST:DATA?**

:CALCulate{1-16}:PARameter{1-16}:MSTatistics:DATA2?

Description: Query only. Outputs the marker statistics of the lower display on the given trace. A dual display trace must be set up before sending the query. The statistics are determined for the entire sweep range. The three values output are Mean (Mean), Standard Deviation (Std Dev), and Peak-to-Peak (P-P), each measured in dB. The values are measured for the entire sweep range for the indicated trace. Markers do not need to be on to output (or display) the statistics.

Cmd Parameters: NA

Query Parameters: <NR3>, <NR3>, <NR3> Outputs marker on-screen statistics.

Range: NA

Default Value: NA

Syntax Example: **:CALC:PAR:MST:DATA2?**

5-18 :CALCulate{1-16}:PARameter{1-16}:SELect Subsystem

The :CALCulate{1-16}:PARameter{1-16}:SELect subsystem command sets the active trace.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:PARameter{1-16}:SELect

Description: Sets the active trace. No query.

To determine the active trace, use:

```
:CALCulate{1-16}:PARameter:SELect?
```

Cmd Parameters: NA

Range: NA

Default Value: 1

Syntax Example: **:CALC:PAR:SEL**

5-19 :CALCulate{1-16}:POLar Subsystem

The :CALCulate{1-16}:POLar subsystem commands configure the polar chart trace displays on a per-channel basis.d and how they are displayed on each trace.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}:POLar Subsystem" on page 5-56
- ":CALCulate{1-16}:PROCessing:ORDer Subsystem" on page 5-58
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}:POLar:ANGLe:STARt <NRf>
:CALCulate{1-16}:POLar:ANGLe:STARt?

Description: The command sets the start angle to use on all polar chart displays on the indicated channel.

The query outputs the start angle to use on all polar chart displays on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Degrees.

Query Parameters: <NR3> The output parameter is in Degrees.

Range: -3.6E2 to 3.6E2

Default Value: 0.00000000000E+000

Syntax Example: **:CALC1:POL:ANGL:STAR 2.10E1**

:CALC1:POL:ANGL:STAR?

:CALCulate{1-16}:POLar:ANGLe:STOP <NRf>
:CALCulate{1-16}:POLar:ANGLe:STOP?

Description: The command sets the stop angle to use on all polar chart displays on the indicated channel. The query outputs the stop angle to use on all polar chart displays on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Degrees.

Query Parameters: <NR3> The output parameter is in Degrees.

Range: -3.6E2 to 3.6E2

Default Value: 3.60000000000E+002

Syntax Example: **:CALC1:POL:ANGL:STOP 2.10E1**

:CALC1:POL:ANGL:STOP?

:CALCulate{1-16}:POLar:CHARt <char>
:CALCulate{1-16}:POLar:CHARt?

Description: The command sets the polar chart mode of all polar chart displays on the indicated channel. The query outputs the polar chart mode of all polar chart displays on the indicated channel.

Cmd Parameters: <char> MAGPhase | MAGSweep

Query Parameters: <char> MAGP | MAGS

Range: NA

Default Value: MAGP

Syntax Example: **:CALC1:POL:CHAR MAGS**

:CALC1:POL:CHAR?

5-20 :CALCulate{1-16}:PROCessing:ORDer Subsystem

The :CALCulate{1-16}:PROCessing:ORDer subsystem configures the measurement port-processing order for the reference plane and group delay.

Time Domain, Group Delay, and Reference Plane Subsystems

Related time domain, group delay, and reference place subsystems are:

- ":CALCulate{1-16}:PROCessing:ORDer Subsystem" on page 5-58
- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":CALCulate{1-16}[:SELected]:TRANsform:TIME Subsystem" on page 5-98
- ":SENSe{1-16}:CORRection:EXTension Subsystem" on page 5-206

:CALCulate{1-16}:PROCessing:ORDer:GRPDelay <char>
:CALCulate{1-16}:PROCessing:ORDer:GRPDelay?

Description: The command sets the order of processing of group delay and trace math data.

The query outputs the order of processing of group delay and trace math data.

Cmd Parameters: <char> GRPDtracemath | TRACemathgrpd

Query Parameters: <char> GRPD | TRAC

Range: NA

Default Value: NA

Syntax Example: **:CALC1:PROC:ORD:GRPD TRAC**

:CALC1:PROC:ORD:GRPD?

:CALCulate{1-16}:PROCessing:ORDer:REFPlane <char>
:CALCulate{1-16}:PROCessing:ORDer:REFPlane?

Description: The command sets the order of processing of reference plane and impedance transformation data.

The query outputs the order of processing of reference plane and impedance transformation data.

Cmd Parameters: <char> IMPedrefplane | REFplaneimped

Query Parameters: <char> IMP | REF

Range: NA

Default Value: NA

Syntax Example: **:CALC1:PROC:ORD:REFP IMP**

:CALC1:PROC:ORD:REFP?

5-21 :CALCulate{1-16}:REFerence Subsystem

The :CALCulate{1-16}:REFerence subsystem commands configure various parameters related to the reference plane in line types such as coaxial, microstrip, and waveguides.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

Time Domain, Group Delay, and Reference Plane Subsystems

Related time domain, group delay, and reference place subsystems are:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":CALCulate{1-16}[:SELected]:TRANsform:TIME Subsystem" on page 5-98
- ":SENSe{1-16}:CORRection:EXTension Subsystem" on page 5-206

:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric <char>
:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric?

Description: Sets the reference plane extension coaxial line dielectric type.

The query outputs the reference plane extension coaxial line dielectric type.

Cmd Parameters: <char> AIR | MICROporous | OTHER | POLYethylene | TEFLON

Query Parameters: <char> AIR | MICRO | OTHER | POLY | TEFLON

Range: NA

Default Value: AIR

Syntax Example:
```
:CALC:REF:EXT:COAX:DIEL AIR
:CALC:REF:EXT:COAX:DIEL?
```

:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric:OTHer <NRf>
:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric:OTHer?

Description: Sets the reference plane extension coaxial line manual dielectric value.

The query outputs the reference plane extension coaxial line manual dielectric value.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: 1 to 9.99E3

Default Value: 1.00000000000E+000

Syntax Example:
```
:CALC:REF:EXT:COAX:DIEL:OTH 1.0E3
:CALC:REF:EXT:COAX:DIEL:OTH?
```

:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric:VALue?

Description: Query only.

The query outputs the reference plane extension coaxial line dielectric value.

Cmd Parameters: NA

Query Parameters: <NR3> The output parameter is a unitless number.

Range: NA

Default Value: 1.00064900000E+000

Syntax Example: **:CALC:REF:EXT:COAX:DIEL:VAL?**

:CALCulate{1-16}:REFerence:EXTension:LINE <char>
:CALCulate{1-16}:REFerence:EXTension:LINE?

Description: Sets the reference plane extension line type. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters. The query outputs the reference plane extension line type.

Cmd Parameters: <char> COAXial

Query Parameters: <char> COAX

Range: NA

Default Value: COAX

Syntax Example: **:CALC:REF:EXT:LINE COAX**

:CALC:REF:EXT:LINE?

:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:AUTOmatic

Description: Calculates and applies the reference plane extension delay for the indicated port. No query.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:REF:EXT:PORT1:AUTO**

:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:DISTance <NRf>
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:DISTance?

Description: Sets the reference plane extension distance in meters for the indicated port. The query outputs the reference plane extension distance in meters for the indicated port.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:REF:EXT:PORT1:DIST 5E-4**

:CALC:REF:EXT:PORT1:DIST?

:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:LOSS <NRf>
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:LOSS?

Description: Sets the reference plane extension loss in dB for the indicated port. The query outputs the reference plane extension loss in dB for the indicated port.

Cmd Parameters: <NRf> The input parameter is in dB.

Query Parameters: <NR3> The output parameter is in dB.

Range: -1E3 to +1E3

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:REF:EXT:PORT1:LOSS 3E0**

:CALC:REF:EXT:PORT1:LOSS?

:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:PHAse <NRf>
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:PHAse?

Description: Sets the reference plane extension phase offset in degrees for the indicated port. The allowable range is -360 degrees to +360 degrees. If the -360 value is exceeded, the instrument truncates the input to -360; if the +360 value is exceeded, the instrument truncates the input to +360. The query outputs the reference plane extension phase offset in degrees for the indicated port.

Cmd Parameters: <NRf> The input parameter is in Degrees.

Query Parameters: <NR3> The output parameter is in Degrees.

Range: -360 to +360

Default Value: 0.000000E+000

Syntax Example: **:CALC:REF:EXT:PORT1:PHA 1.5E1**

:CALC:REF:EXT:PORT1:PHA?

:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:TIMe <NRf>
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:TIMe?

Description: Sets the reference plane extension in time in seconds for the indicated port. The query outputs the reference plane extension in time in seconds for the indicated port.

Cmd Parameters: <NRf> The input parameter is in Seconds.

Query Parameters: <NR3> The output parameter is in Seconds.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:REF:EXT:PORT1:TIM 5.0E-2**

:CALC:REF:EXT:PORT1:TIM?

:CALCulate{1-16}:RFRanging: <char>

Description: Sets the RF ranging to On for measurement.

Cmd Parameters: ENABLE | DISABLE

Query Parameters: NA

Range: NA

Default Value: ENABLE

Syntax Example: **:CALC:RFR:ENABLE**

5-22 :CALCulate{1-16}[:SELected]:CONVersion Subsystem

The :CALCulate{1-16}[:SELected]:CONVersion subsystem sets the parameter conversion configuration and control for the indicated channel and the active trace.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}:POLar Subsystem" on page 5-56
- ":CALCulate{1-16}:PROCessing:ORDer Subsystem" on page 5-58
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:CONVersion:FUNCtion <char>
:CALCulate{1-16}[:SELected]:CONVersion:FUNCtion?

Description: The command sets the parameter conversion function on the active trace of the indicated channel. The query outputs the parameter conversion function on the active trace of the indicated channel.

Cmd Parameters: <char> ZREFlection | ZTRansmit | YREFlection | YTRansmit | INVersion

Where:

- ZREFlection = Sets the conversion parameter to Z:Reflection.
- ZTRansmit = Sets the conversion parameter to Z:Transmission.
- YREFlection = Sets the conversion parameter to Y:Reflection.
- YTRansmit = Sets the conversion parameter to Y:Transmission.
- INVersion = Inverts the parameter to 1/S.

Query Parameters: <char> ZREF | ZTR | YREF | YTR | INV

Range: NA

Default Value: INV

Syntax Example:
```
:CALC1:CONV:FUNC ZREF
:CALC1:CONV:FUNC?
```

:CALCulate{1-16}[:SELected]:CONVersion[:STATe] <char>
:CALCulate{1-16}[:SELected]:CONVersion[:STATe]?

Description: The command sets the on/off status of parameter conversion on the active trace of the indicated channel. The query outputs the on/off status of parameter conversion on the active trace of the indicated channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Default Value: 0

Syntax Example:
```
:CALC1:CONV ON
:CALC1:CONV?
```

5-23 :CALCulate{1-16}[:SELected]:DATA Subsystem

The :CALCulate{1-16}[:SELected]:DATA subsystem commands input and output various instrument information sets such as trace data and S-Parameters.

I/O Configuration and File Operation Subsystems

Related subsystems for I/O configuration and file operation are:

- ":CALCulate{1-16}:FORMat Subsystem - SnP Data" on page 5-29
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":FORMat Subsystem" on page 5-122
- ":HCOPy Subsystem" on page 5-125
- ":MMEMory Subsystem" on page 5-130

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}[:SELected]:FORMat Subsystem" on page 5-67
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORMat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:DATA:FDATa <block>
:CALCulate{1-16}[:SELected]:DATA:FDATa?

Description: Inputs formatted data to display on the active trace. The instrument must be in Hold Sweep. The query outputs formatted data of the active trace.

Cmd Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:DATA:FDAT <block>**

:CALC:DATA:FDAT?

:CALCulate{1-16}[:SELected]:DATA:FMEMory <block>
:CALCulate{1-16}[:SELected]:DATA:FMEMory?

Description: Inputs formatted data to display on the active trace and writes it to trace memory. The command can also perform math functions. The query outputs the formatted memory of the active trace.

Cmd Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Cmd Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:DATA:FMEM <block>**

:CALC:DATA:FMEM?

:CALCulate{1-16}[:SELected]:DATA:SDATa <Block>
:CALCulate{1-16}[:SELected]:DATA:SDATa?

Description: Inputs corrected real and imaginary S-parameter data to display on the active trace. The data must be real and imaginary formatted data. The command converts the active trace display appropriately. The query outputs the real/imaginary S-parameter data for the active trace.

Cmd Parameters: <Block> See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <Block>

Output: <Block>

Range: NA

Default Value: NA

Syntax Example: **:CALC:DATA:SDAT <block>**

:CALC:DATA:SDAT?

:CALCulate{1-16}[:SELected]:DATA:SMEMory <Block>
:CALCulate{1-16}[:SELected]:DATA:SMEMory?

Description: Inputs corrected S-parameter trace memory to display on the active trace. The query outputs corrected S-parameter trace memory of the active trace.

Cmd Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:DATA:SMEM <block>**

:CALC:DATA:SMEM?

5-24 :CALCulate{1-16}[:SELected]:FORmat Subsystem

The :CALCulate{1-16}[:SELected]:FORmat subsystem command configures the display type for the active trace.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:FORMat <char>
:CALCulate{1-16}[:SELected]:FORMat?

Description: Selects the display format of the active trace. The query outputs the display format of the active trace. The available display types are:

- ETAmax = η Max (MEA-only)
- GDELay = Group Delay
- IMAGinary = Imaginary
- KQ = kQ quality (MEA-only)
- KQETAmax = kQ quality and η Max (MEA-only)
- LINPHase = Linear Mag and Phase
- LOGPHase = Log Magnitude and Phase
- MLINear = Linear Magnitude
- MLOGarithmic = Log Magnitude
- PHASe = Phase
- PLINear = Linear Polar and Linear Phase
- PLINCOMPlex = Linear Polar and Real/Imaginary
- PLOGarithmic = Log Polar and Log Phase
- PLOGCOMPlex = Log Polar and Real/Imaginary
- ZREAL = Impedance Real
- ZCAPacitance = Impedance Capacitance
- ZCOMPlex = Impedance Real and Imaginary
- ZIMAGinary = Impedance Imaginary
- ZINDuctance = Impedance Inductance
- ZMAGNitude = Impedance Magnitude

The query outputs the display format of the indicated trace. See Chapter 2, "Programming the ShockLine™ Series VNA" for more information and a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <char> ETAmax |GDELay | IMAGinary | KQ | KQETAmax | LINPHase | LOGPHase | MLINear | MLOGarithmic | PHASe | PLINear | PLINCOMPlex | PLOGarithmic | PLOGCOMPlex | REAL | REIMaginary | SCOMPlex | SLINear | SLOGarithmic | SMITh | SWR | ZREAL |ZCAPacitance | ZCOMPlex | ZIMAGinary | ZINDuctance | ZMAGNitude

Query Parameters: <char> ETAmax | ETA | GDEL | IMAG | KQ | KQETA | LINPH | LOGPH | MLIN | MLOG | PHAS | PLIN | PLINCOMP | PLOG | PLOGCOMP | REAL | REIM | SCOMP | SLIN | SLOG | SMIT | SWR | ZREAL |ZCAP | ZCOMP | ZIMAG |ZIND | ZMAGN

Range: NA

Default Value: MLOG for all traces

Syntax Example: **:CALC:FORM LOGPH**

:CALC:FORM?

:CALCulate{1-16}[:SELected]:IMPedance:LC[:STATe] <char>
:CALCulate{1-16}[:SELected]:IMPedance:LC[:STATe]?

Description: : Toggles the display of marker LC value on impedance measurements of the active trace on/off.

Effects only apply to PLINear, PLINCOMPlex, PLOGarithmic, PLOGCOMPlex ,and SMITH graph types.

The query returns the status of the marker LC value display on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: :CALC:IMP:LC ON

:CALC:IMP:LC?

5-25 :CALCulate{1-16}[:SELected]:LIMit Subsystem

The :CALCulate{1-16}[:SELected]:LIMit subsystem provides limit line configuration and control for the active trace.

Limit Line and Segment Subsystems

Related limit line and segment configuration and control subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107
- ":SENSe{1-16}:FSEGMent Subsystem" on page 5-211.
- ":SENSe{1-16}:FSEGMent{1-50} Subsystem" on page 5-218.
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231
- ":SENSe{1-16}:SEGMent Subsystem" on page 5-235

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:LIMit:ALARm <char>
:CALCulate{1-16}[:SELected]:LIMit:ALARm?

Description: Turns the limit alarm function on/off for the whole system. The query outputs the limit alarm on/off status of the system.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:LIM:ALAR ON**

:CALC:LIM:ALAR?

:CALCulate{1-16}[:SELected]:LIMit:DATA <block>
:CALCulate{1-16}[:SELected]:LIMit:DATA?

Description: Inputs the limit line table for the active trace. Outputs the limit line table of the active trace.

Cmd Parameters: <block> Block data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> Block data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:LIM:DATA <block>**

:CALC:LIM:DATA?

:CALCulate{1-16}[:SELected]:LIMit:DISPlay[:STATe] <char>
:CALCulate{1-16}[:SELected]:LIMit:DISPlay[:STATe]?

Description: Turns limit display on/off for the active trace. The query outputs the limit display on/off status for the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:LIM:DISP ON**

:CALC:LIM:DISP?

:CALCulate{1-16}[:SELected]:LIMit:FAIL?

Description: Query only. The query outputs the limit testing result for the active trace, where:

- "0" = The limit passed.
- "1" = The limit failed.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 | 1

Default Value: 0

Syntax Example: **:CALC:LIM:FAIL?**

:CALCulate{1-16}[:SELected]:LIMit:OFF

Description: Turns all limits off and clears limit table. No query.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:LIM:OFF**

:CALCulate{1-16}[:SELected]:LIMit:REPort?

Description: Query only. The query outputs a list of formatted data points that failed the limit test on the active trace. Each failed point is listed on a separate line as:

`LIMTYPE, FREQ, YVAL, LIMYVAL`

- `LIMTYPE` = UPPER, LOWER, or BOTH
- `FREQ` = Frequency of failing point
- `YVAL` = Y value of the failing point
- `LIMYVAL` = Limit Y value

Cmd Parameters: NA

Query Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **`:CALC:LIM:REP?`**

`#9000000638UPPER, 9.20885000000E+008, 9.874321E+000, 9.800000E+000`

:CALCulate{1-16}[:SELected]:LIMit:REPort:POINt?

Description: Query only. The query outputs the number of points failing limit testing.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to the current number of set measurement points.

Default Value: 0

Syntax Example: **`:CALC:LIM:REP:POIN?`**

:CALCulate{1-16}[:SELected]:LIMit:SEGMent:ADD {No argument} | {<char>} | {<char>,<NRf>,<NRf>}

Description: Adds a limit line segment for the active trace. If the optional parameters are omitted, an empty segment is added. No query.

Limit Lines for Rectilinear Displays

For rectilinear displays, up to 50 segment lines can be added to each trace display.

Cmd Parameters: {<No argument>} | {<char>} | {<char>,<NRf>,<NRf>}

<No argument> If no argument is added to the command, the command adds an empty segment.

<char> NONE | UPPer | LOWER

<NRf> Start time, frequency, or distance for X1.

<NRf> Stop time, frequency, or distance for X2.

Range: NA

Default Value: NA

Syntax Example: **`:CALC:LIM:SEGM:ADD UPPER, 2.0E9, 3.0E9`**

:CALCulate{1-16}[:SELected]:LIMit:SEGMent:CLEar

Description: Clears all the limit segment definitions on the active trace. No query.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:LIM:SEGM:CLE**

:CALCulate{1-16}[:SELected]:LIMit:SEGMent:COUNt?

Description: Query only. Outputs the number of limit segments defined on the active trace.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 50

Default Value: 0

Syntax Example: **:CALC:LIM:SEGM:ADD UPPER, 2.0E9, 3.0E9**

:CALC:LIM:SEGM:COUN?

:CALCulate{1-16}[:SELected]:LIMit:SEGMent:DEFine <NRf>,<NRf> | <NRf>,<NRf>,<NRf>,<NRf>
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:DEFine?

Description: Defines a limit line segment for an added segment for the active trace. Output the current limit line segment for the active trace.

- If two <NRf> parameters are used, it defines the Y1 start value and the Y2 stop value.
- If four <NRf> parameters are used, it defines the Y1 start value and the Y2 stop value for the upper trace in a dual trace display, and the Y12 (Y1sub in the GUI) start value and the Y22 (Y2sub in the GUI) stop value for the lower trace in a dual trace display.

Cmd Parameters: <NRf>,<NRf> | <NRf>,<NRf>,<NRf>,<NRf>

Query Parameters: <NRf>,<NRf> | <NRf>,<NRf>,<NRf>,<NRf>

Range: NA

Default Value: NA

Syntax Example: **:CALC:LIM:SEGM:DEF 2.0, 3.0**

:CALC:LIM:SEGM:DEF?

:CALCulate{1-16}[:SELected]:LIMit[:STATe] <char>
:CALCulate{1-16}[:SELected]:LIMit[:STATe]?

Description: The command turns limit testing on/off for the active trace of the indicated channel. The query outputs the limit testing on/off status for the active trace of the given channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC1:LIM ON**

:CALC1:LIM?

:CALCulate{1-16}[:SELected]:LIMit:SEGMent:TYPe <char>
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:TYPe?

Description: Sets the limit line type of the current limit line segment being defined for the active trace of the indicated channel where:

- UPPer = Upper limit line
- LOWer = Lower limit line
- NONe = No limit line

Outputs the limit line type of the current limit line segment being defined for the active trace of the indicated channel.

Note: The limit line must exist before using this command. If no limit lines have been created, use the prerequisite command below to add one or more limit lines:

:CALCulate{1-16}:LIMit:SEGMent:ADD

This adds one (1) limit segment which permits the example below to work.

Cmd Parameters: <char> UPPer | LOWer | NONe

Query Parameters: NA

Output: <char> UPP | LOW | NON

Range: NA

Default Value: NON

Syntax Example: **:CALC1:LIM:SEGM:TYP UPP**

:CALC1:LIM:SEGM:TYP?

5-26 :CALCulate{1-16}[:SELected]:MARKer Subsystem

The :CALCulate{1-16}[:SELected]:MARKer subsystem commands control the active marker display, value, and search functions.

Marker Subsystems

Related marker configuration, control, and reporting commands are described multiple subsystems:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:MARKer:ACTivate?

Description: Query only. Outputs the active marker number of the active trace.

To activate a marker, use the command:

```
:CALCulate{1-16}[:SELected]:MARKer{1-13}:ACTivate
```

Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 13

Default Value: 0

Syntax Example: **:CALC:MARK:ACT?**

:CALCulate{1-16}[:SELected]:MARKer:ALL[:STATe] <char>

Description: Toggles all markers on/off on the active trace. No query.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:ALL ON**

:CALCulate{1-16}[:SELected]:MARKer:MOVe:CENTer

Description: Moves the active marker range value to the stimulus center range on the active trace. No query.

To query current location of the active marker, use:

```
:CALCulate{1-16}:MARKer{1-13}:X?
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:MOV:CENT**

:CALCulate{1-16}[:SELected]:MARKer:MOVe:REFMarker

Description: Moves the active marker to the reference marker on the active trace. The active marker cannot be a reference marker. The reference marker must be on. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:MOV:REFM**

:CALCulate{1-16}[:SELected]:MARKer:MOVe:STARt

Description: Moves the active marker range value to the stimulus start range on the active trace. No query.

To query current location of the active marker, use:

```
:CALCulate{1-16}:MARKer{1-13}:X?
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:MOV:STAR**

:CALCulate{1-16}[:SELected]:MARKer:MOVe:STOP

Description: Moves the active marker range value to the stimulus stop range on the active trace. No query.

To query current location of the active marker, use:

```
:CALCulate{1-16}:MARKer{1-13}:X?
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:MOV:STOP**

:CALCulate{1-16}[:SELected]:MARKer:MPSEArch

Description: Performs a multiple peak search on the active trace. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:MPSEA**

:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:EXCursion <NRf>
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:EXCursion?

Description: Sets the marker search excursion value for multiple peak searches on the active trace. The query outputs the marker search excursion value for multiple peak searches on the active trace.

Cmd Parameters: <NRf> The input parameter is in dB.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:MPSEA:EXC 1.8E1**

:CALC:MARK:MPSEA:EXC?

:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:POLarity <char>
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:POLarity?

Description: Sets the marker search polarity value for multiple peak searches on the active trace. The query outputs the marker search polarity value for multiple peak searches on the active trace.

Cmd Parameters: <char> POSitive | NEGative | BOTH

Query Parameters: <char> POS | NEG | BOTH

Range: NA

Default Value: POS

Syntax Example: **:CALC:MARK:MPSEA:POL POS**

:CALC:MARK:MPSEA:POL?

:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:THREshold <NRf>
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:THREshold?

Description: Sets the marker search threshold value for multiple peak searches on the active trace. The query outputs the marker search threshold value for multiple peak searches on the active trace.

Cmd Parameters: <NRf> The input parameter is in dB.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:MPSEA:THRE 1.8E1**

:CALC:MARK:MPSEA:THRE?

:CALCulate{1-16}[:SELected]:MARKer:MTSEArch

Description: Performs a multiple target search on the active trace. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:MTSEA**

:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TARget <NRf>
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TARget?

Description: Sets the marker search target value for multiple target searches on the active trace. The query outputs the marker search target value for multiple target searches on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3>, <NR3>, <NR3> The output parameter depends on the display type.

Range: MPNI

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:MTSEA:TAR 1E7**

:CALC:MARK:MTSEA:TAR?

:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TRANsition <char>
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TRANsition?

Description: Sets the marker search transition value for multiple target searches on the active trace. The query outputs the marker search transition value for multiple target searches on the active trace.

Cmd Parameters: <char> POSitive | NEGative | BOTH

Query Parameters: <char> POS | NEG | BOTH

Range: NA

Default Value: BOTH

Syntax Example: **:CALC:MARK:MTSEA:TRAN POS**

:CALC:MARK:MTSEA:TRAN?

:CALCulate{1-16}[:SELected]:MARKer:OFF

Description: Turns all markers off. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:OFF**

:CALCulate{1-16}[:SELected]:MARKer:PSEArch <char>
:CALCulate{1-16}[:SELected]:MARKer:PSEArch?

Description: Performs a peak search on the active trace. The query outputs the last marker search type for peak searches on the active trace.

Cmd Parameters: <char> PEAK | LEFT | RIGHT

Query Parameters: <char> PEAK | LEFT | RIGHT

Range: NA

Default Value: PEAK

Syntax Example: **:CALC:MARK:PSEA PEAK**

:CALC:MARK:PSEA?

:CALCulate{1-16}[:SELected]:MARKer:PSEArch:EXCursion <NRf>
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:EXCursion?

Description: Sets the marker search excursion value for peak searches on the active trace. The query outputs the marker search excursion value for peak searches on the active trace.

Cmd Parameters: <NRf> The input parameter is in dB.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: MPNI

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:PSEA:EXC 1.8E1**

:CALC:MARK:PSEA:EXC?

:CALCulate{1-16}[:SELected]:MARKer:PSEArch:POLarity <char>
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:POLarity?

Description: Sets the marker search polarity value for peak searches on the active trace. The query outputs the marker search polarity value for peak searches on the active trace.

Cmd Parameters: <char> POSitive | NEGative | BOTH

Query Parameters: <char> POS | NEG | BOTH

Range: NA

Default Value: POS

Syntax Example: **:CALC:MARK:PSEA:POL POS**

:CALC:MARK:PSEA:POL?

:CALCulate{1-16}[:SELected]:MARKer:PSEArch:THREshold <NRf>
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:THREshold?

Description: Sets the marker search threshold value for peak searches on the active trace. The query outputs the marker search threshold value for peak searches on the active trace.

Cmd Parameters: <NRf> The input parameter is in dB.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:PSEA:THRE 1.8E1**

:CALC:MARK:PSEA:THRE?

:CALCulate{1-16}[:SELected]:MARKer:SEArch <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch?

Description: Sets the marker search type for the active marker on the active trace. The query outputs the marker search type for the active marker on the active trace.

Cmd Parameters: <char> MAX | MIN | PEAK | TARGet

Query Parameters: <char> MAX | MIN | PEAK | TARG

Range: NA

Default Value: MAX

Syntax Example: **:CALC:MARK:SEA MAX**

:CALC:MARK:SEA?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:DATA?

Description: Query only. The query outputs the marker search bandwidth data of the active marker on the active trace.

Query Parameters: <NR3>, <NR3>, <NR3>, <NR3>{,<NR3>} The command outputs four or five <NR3> data elements in the following sequence:

- First <NR3> - Bandwidth in Hertz.
- Second <NR3> - Center of marker search range in Hertz, Meters, or Seconds.
- Third <NR3> - Q which is a unitless number.
- Fourth <NR3> - Loss in dB.
- Fifth <NR3> - Optional. The Shape Factor which is a unitless number. This parameter is optional depending on if the Shape Function is turned on or off using the Shape Command.

Syntax Example: **:CALC:MARK:SEA:BAND:DATA?**

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:DEFine <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:DEFine?

Description: The command sets the marker search define value for bandwidth calculation of the active marker on the active trace. The query outputs the marker search define value for bandwidth calculation of the active marker on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:SEA:BAND:DEF 3.0E4**

:CALC:MARK:SEA:BAND:DEF?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:RTYPe <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:RTYPe?

Description: The command sets the reference type for bandwidth search on the active trace of the indicated channel. The query outputs the reference type for bandwidth search on the active trace of the indicated channel.

Cmd Parameters: MARKer | RVALue

- MARKer = Marker
- RVALue = Reference Value

Query Output: MARK | RVAL

Range: NA

Default Value: MARK

Syntax Example: **:CALC1:MARK:SEA:BAND:RTYPE RVAL**

:CALC1:MARK:SEA:BAND:RTYPE?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:RVALue?

Description: The command sets the reference value for bandwidth search on the active trace of the indicated channel. The query outputs the reference value for bandwidth search on the active trace of the indicated channel.

Cmd Parameters: <NRf>

Query Output: <NR3>

Range: NA

Default Value: 0

Syntax Example: **:CALC1:MARK:SEA:BAND:RVAL 5**

:CALC1:MARK:SEA:BAND:RVAL?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:HIGH <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:HIGH?

Description: The command sets the marker search high value for bandwidth shape factor calculation of the active marker on the active trace. The query outputs the marker search high value for bandwidth shape factor calculation of the active marker on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:SEA:BAND:SHAP:HIGH 4E0**

:CALC:MARK:SEA:BAND:SHAP:HIGH?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:LOW <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:LOW?

Description: The command sets the marker search low value for bandwidth shape factor calculation of the active marker on the active trace. The query outputs the marker search low value for bandwidth shape factor calculation of the active marker on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:SEA:BAND:SHAP:LOW 4E0**

:CALC:MARK:SEA:BAND:SHAP:LOW?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe[:STATe]?

Description: The command toggles on/off the marker search bandwidth shape factor calculation of the active marker on the active trace. The query outputs the on/off status of marker search bandwidth shape factor of the active marker calculation on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:BAND:SHAP 1**

:CALC:MARK:SEA:BAND:SHAP?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SSTart?

Description: The command sets the bandwidth start start position on the active trace of the indicated channel. This is only use when the bandwidth reference type is set to reference value. The query outputs the bandwidth start start position on the active trace of the indicated channel.

Cmd Parameters: BEGinning|MAXimum

- BEGinning

- MAXimum = Reference

Query Parameters: <NR3>

Query Output: BEG|MAX

Range: NA

Default Value: MAX

Syntax Example: **:CALC1:MARK:SEA:BAND:SSTart MAX**

:CALC1:MARK:SEA:BAND:SSTart?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth[:STATe]?

Description: The command toggles on/off the marker search bandwidth calculation of the active marker on the active trace. The query outputs the on/off status of marker search bandwidth calculation of the active marker on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:BAND ON**

:CALC:MARK:SEA:BAND?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:DATA?

Description: Query only. The query outputs the marker search notch bandwidth data of the active marker on the active trace.

Query Parameters: <NR3>, <NR3>, <NR3>, <NR3>{, <NR3>} The output parameter is in Hertz, Meters, or Seconds.

Range: Outputs multiple NR3 data

Default Value: NA

Syntax Example: **:CALC:MARK:SEA:NOTC:DATA?**

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:DEFine <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:DEFine?

Description: The command sets the marker search define value for notch bandwidth calculation of the active marker on the active trace. The query outputs the marker search define value for notch bandwidth calculation of the active marker on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:SEA:NOTC:DEF 1.0E4**

:CALC:MARK:SEA:NOTC:DEF?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RTYPe <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RTYPe?

Description: The command sets the reference type for notch search on the active trace of the indicated channel. The query outputs the reference type for notch search on the active trace of the indicated channel.

MARKer = Marker reference

RVALue = Reference Value entered into the Reference Value toolbar.

Cmd Parameters: MARKer|RVALue

Query Output: MARK|RVAL

Range: N/A

Default Value: MARK

Syntax Example: **:CALC1:MARK:SEA:NOTCh:RTYPE RVAL**

Syntax Example: **:CALC1:MARK:SEA:NOTCh:RTYPE?**

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RVALue <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RVALue?

Description: The command sets the reference value for notch search on the active trace of the indicated channel. The query outputs the reference value for notch search on the active trace of the indicated channel.

Cmd Parameters: <NRf>

Query Output: <NR3>

Range: N/A

Default Value: 0

Syntax Example: **:CALC1:MARK:SEA:NOTCh:RVAL 5**

Syntax Example: **:CALC1:MARK:SEA:NOTCh:RVAL?**

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:HIGH <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:HIGH?

Description: The command sets the marker search high value for notch shape factor calculation of the active marker on the active trace. The query outputs the marker search high value for notch shape factor calculation of the active marker on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:SEA:NOTC:SHAP:HIGH 4E0**

:CALC:MARK:SEA:NOTC:SHAP:HIGH?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:LOW <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:LOW?

Description: The command sets the marker search low value for notch shape factor calculation of the active marker on the active trace. The query outputs the marker search low value for notch shape factor calculation of the active marker on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:SEA:NOTC:SHAP:LOW 4E0**

:CALC:MARK:SEA:NOTC:SHAP:LOW?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe[:STATe]?

Description: The command toggles on/off the marker search notch shape factor calculation of the active marker on the active trace. Outputs on/off status of marker search notch shape factor calculation of the active marker on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:NOTC:SHAP ON**

:CALC:MARK:SEA:NOTC:SHAP?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SSTart <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SSTart?

Description: The command sets the notch start start position on the active trace of the indicated channel. This is only use when the notch reference type is set to reference value. The query outputs the notch start position on the active trace of the indicated channel.

Cmd Parameters: Cmd parameters: BEGinning|MINimum

- BEGinning
- MINimum = Reference

Query Output: BEG|MIN

Range: N/A

Default Value: MIN

Syntax Example: **:CALC1:MARK:SEA:NOTCh:SSTart MIN**

:CALC1:MARK:SEA:NOTCh:SSTart?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh[:STATe]?

Description: The command toggles on/off the marker search notch calculation of the active marker on the active trace. The query outputs the on/off status of marker search notch calculation of the active marker on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:NOTC ON**

:CALC:MARK:SEA:NOTC?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:ALL[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:ALL[:STATe]?

Description: Toggles on/off applying the marker search range to all traces. The query outputs the on/off status of applying the marker search range to all traces.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:RANG:ALL ON**

:CALC:MARK:SEA:RANG:ALL?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STARt:X <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STARt:X?

Description: Sets the marker search range start range value on the active trace. The query outputs the marker search range start range value on the active trace.

Cmd Parameters: <NRf> The input parameter is in Hertz, Meters, or Seconds.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:MARK:SEA:RANG:STAR:X 2.5E9**

:CALC:MARK:SEA:RANG:STAR:X?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STOP:X <NRf>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STOP:X?

Description: Sets the marker search range stop range value on the active trace. The query outputs the marker search range stop range value on the active trace.

Cmd Parameters: <NRf> The input parameter is in Hertz, Meters, or Seconds.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:MARK:SEA:RANG:STOP:X 2.5E9**

:CALC:MARK:SEA:RANG:STOP:X?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe[:STATe]?

Description: Toggles on/off the marker search range on the active trace. The query outputs the on/off status of marker search range on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:RANG ON**

:CALC:MARK:SEA:RANG?

:CALCulate{1-16}[:SELected]:MARKer:SEArch:TRACKing[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer:SEArch:TRACKing[:STATe]?

Description: Toggles on/off marker search tracking on the active trace. The query outputs the on/off status of marker search tracking on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:MARK:SEA:TRACK ON**

:CALC:MARK:SEA:TRACK?

:CALCulate{1-16}[:SELected]:MARKer:SET:CENTer

Description: Sets the stimulus center range to the active marker range value on the active trace. Equivalent to setting `:SENSe{1-16}:FREQuency:CENTer` to the active marker X value. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:SET:CENT**

:CALCulate{1-16}[:SELected]:MARKer:SET:REFLevel

Description: Sets the display reference level to the active marker response value on the active trace. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:SET:REFL**

Related Cmds: :DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV <NRf> on page 5-119

:CALCulate{1-16}[:SELected]:MARKer:SET:STARt

Description: Sets the stimulus start range to the active marker range value on the active trace. Equivalent to setting :SENSe{1-16}:FREQuency:STARt to the active marker X value. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:SET:STAR**

:CALCulate{1-16}[:SELected]:MARKer:SET:STOP

Description: Sets the stimulus stop range to the active marker range value on the active trace. Equivalent to setting :SENSe{1-16}:FREQuency:STOP to the active marker X value. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK:SET:STOP**

:CALCulate{1-16}[:SELected]:MARKer:TSEArch <char>
:CALCulate{1-16}[:SELected]:MARKer:TSEArch?

Description: Performs a target search on the active trace. The query outputs the marker search type for target searches on the active trace.

Cmd Parameters: <char> TARGet | LEFT | RIGHT

Query Parameters: <char> TARG | LEFT | RIGHT

Range: NA

Default Value: TARG

Syntax Example: **:CALC:MARK:TSEA TARG**

:CALC:MARK:TSEA?

:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TARget <NRf>
:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TARget?

Description: Sets the marker search target value for target searches on the active trace. The query outputs the marker search target value for target searches on the active trace.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3>, <NR3>, <NR3> The output parameter depends on the display type.

Range: MPNF

Default Value: 0.000000E+000

Syntax Example: **:CALC:MARK:TSEA:TAR 1E7**

:CALC:MARK:TSEA:TAR?

:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TRANsition <char>
:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TRANsition?

Description: Sets the marker search transition value for target searches on the active trace. The query outputs the marker search transition value for target searches on the active trace.

Cmd Parameters: <char> POSitive | NEGative | BOTH

Query Parameters: <char> POS | NEG | BOTH

Range: NA

Default Value: BOTH

Syntax Example: **:CALC:MARK:TSEA:TRAN POS**

:CALC:MARK:TSEA:TRAN?

5-27 :CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem

The :CALCulate{1-16}[:SELected]:MARKer{1-13} subsystem commands provide configuration and control for the indicated marker.

Marker Subsystems

Related marker configuration, control, and reporting commands are described multiple subsystems:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:MARKer{1-13}:ACTivate

Description: Makes the indicated marker of the active trace the active marker. No query.

To find if a marker is active, use the query:

```
:CALCulate{1-16}[:SELected]:MARKer:ACTivate?
```

Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

Cmd Parameters: NA

Range: NA

Default Value: 0

Syntax Example: **`:CALC:MARK1:ACT`**

:CALCulate{1-16}[:SELected]:MARKer{1-13}:MOVe <char>

Description: Moves the indicated marker to the indicated location on the active trace. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker. No query.

The available locations are:

- CENTer = Moves the marker to the center of the trace range.
- REFmarker = Moves the marker to the reference marker on the active trace

- STARt = Moves the marker to the start frequency of the active trace
- STOP = Moves the marker to the stop frequency of the active trace

Cmd Parameters: <char> CENTer | REFmarker | STARt | STOP

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK1:MOV CENT**

:CALCulate{1-16}[:SELected]:MARKer{1-13}:SET <char>

Description: Sets the start, stop, or center range of the display reference level to the indicated marker range/response on the active trace. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker. No query.

Cmd Parameters: <char> CENTer | REFmarker | STARt | STOP

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK1:SET CENT**

:CALCulate{1-16}[:SELected]:MARKer{1-13}:X <NRf>
:CALCulate{1-16}[:SELected]:MARKer{1-13}:X?

Description: Enters the frequency, distance, or time of indicated marker on the active trace and turn on. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker. The query outputs the frequency, distance, or time of the indicated marker on the active trace.

Cmd Parameters: <NRf> The input parameter is in Hertz, Meters, or Seconds.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: The range depends on the parameter type setting:

- Frequency = Minimum Instrument Frequency to the Maximum Instrument Frequency
- Time = 1E-9 Seconds to 4E-9 Seconds
- Distance = -29.965E-3 Meters to 1.1988 Meters

Default Value: 10 MHz

Syntax Example: **:CALC:MARK1:X 2.0E7**

:CALC:MARK1:X?

:CALCulate{1-16}[:SELected]:MARKer{1-13}:Y?

Description: Query only. The query outputs the response value of the indicated marker on the active trace.Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

See Table 2-7, “Trace Parameters and Coefficients” on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Query Parameters: <NR3> | <NR3>, <NR3> The output parameters depend on the display type.

Range: NA

Default Value: NA

Syntax Example: **:CALC:MARK1:Y?**

:CALCulate{1-16}[:SELected]:MARKer{1-13}[:STATe] <char>
:CALCulate{1-16}[:SELected]:MARKer{1-13}[:STATe]?

Description: Toggles on/off displaying the indicated marker of the active trace on/off. Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker. Outputs the on/off display status of the indicated marker of the active trace.Markers 1 through 12 are standard measurement markers. Marker 13 is the reference marker.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 1

Syntax Example: **:CALC:MARK1 ON**

:CALC:MARK1?

5-28 :CALCulate{1-16}[:SELected]:MATH Subsystem

The :CALCulate{1-16}[:SELected]:MATH subsystem commands provide configuration and control for inter-trace mathematics operations.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}:POLar Subsystem" on page 5-56
- ":CALCulate{1-16}:PROCessing:ORDer Subsystem" on page 5-58
- ":CALCulate{1-16}[:SELected]:CONVersion Subsystem" on page 5-63
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MATH Subsystem" on page 5-91
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:MATH:DISPlay <char>
:CALCulate{1-16}[:SELected]:MATH:DISPlay?

Description: The command selects the trace memory display operation for the active trace of the indicated channel where:

- DATA = Display only the current sweep data
- MEM = Display only the memory data
- DTM = Display both the current sweep data and the memory data at the same time.
- DMM = Combine the sweep data and the memory data mathematically using a addition, subtraction, multiplication, or division and display only the results.
- OFF = Turn the trace display.

The query outputs the trace memory display operation for the active trace of the indicated channel.

Cmd Parameters: <char> DATA | MEM | DTM | DMM | OFF

Query Parameters: <char> DATA | MEM | DTM | DMM | OFF

Range: NA

Default Value: DATA

Syntax Example:
```
:CALC1:MATH:DISP MEM
:CALC1:MATH:DISP?
```

:CALCulate{1-16}[:SELected]:MATH:FUNCtion <char>
:CALCulate{1-16}[:SELected]:MATH:FUNCtion?

Description: The command selects the trace memory math operation on the active trace of the indicated channel. The query outputs the trace memory math operation on the active trace of the indicated channel.

Cmd Parameters: <char> ADD | SUBTract | MULTiply | DIVide

Query Parameters: <char> ADD | SUBT | MULT | DIV

Range: NA

Default Value: DIV

Syntax Example: `:CALC1:MATH:FUNC ADD`

`:CALC1:MATH:FUNC?`

:CALCulate{1-16}[:SELected]:MATH:INTErtrace:FUNCtion <char>
:CALCulate{1-16}[:SELected]:MATH:INTErtrace:FUNCtion?

Description: The command selects the inter-trace memory math operation on the active trace of the indicated channel. The result will be displayed on the active trace. The query outputs the inter-trace memory math operation on the active trace of the indicated channel.

Cmd Parameters: <char> ADD | SUBTract | MULTiply | DIVide

Query Parameters: <char> ADD | SUBT | MULT | DIV

Range: NA

Default Value: DIV

Syntax Example: **`:CALC1:MATH:INTE:FUNC ADD`**

`:CALC1:MATH:INTE:FUNC?`

:CALCulate{1-16}[:SELected]:MATH:INTErtrace:OPERand{1-2}:DEFine <char1>, <char2>
:CALCulate{1-16}[:SELected]:MATH:INTErtrace:OPERand{1-2}:DEFine?

Description: The command sets the trace number and data type for the indicated operand on the active trace of the indicated channel. Note that both parameters must be defined.

The <char1> value sets the trace number from the following selections:

- TR1 = Trace 1
- TR2 = Trace 2
- TR3 = Trace 3
- TR4 = Trace 4
- TR5 = Trace 5
- TR6 = Trace 6
- TR7 = Trace 7
- TR8 = Trace 8
- TR9 = Trace 9
- TR10 = Trace 10
- TR11 = Trace 11
- TR12 = Trace 12
- TR13 = Trace 13
- TR14 = Trace 14

- TR15 = Trace 15
- TR16 = Trace 16

The <char2> value sets the data operand as:

- DATA = Display just the current sweep data
- DMM = Combine the sweep data and the memory data mathematically using a addition, subtraction, multiplication, or division and display only the results.

The query outputs the trace number and data type for the indicated operand on the active trace of the indicated channel.

Cmd Parameters: <char1> TR1 | TR2 | TR3 | TR4 | TR5 | TR6 | TR7 | TR8 | TR9 | TR10 | TR11 | TR12 | TR13 | TR14 | TR15 | TR16

<char2> DATA | DMM

Query Parameters: <char1>, <char2>

Range: NA

Default Value: TR1, DATA

Syntax Example: **:CALC1:MATH:INTE:OPER1:DEF TR1, DATA**

:CALC1:MATH:INTE:OPER1:DEF?

:CALCulate{1-16}[:SELected]:MATH:INTErtrace[:STATe] <char>
:CALCulate{1-16}[:SELected]:MATH:INTErtrace[:STATe]?

Description: The command toggles on/off the inter-trace math operation on the active trace of the indicated channel. The query outputs the on/off state of the inter-trace math operation on the active trace of the indicated channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC1:MATH:INTE ON**

:CALC1:MATH:INTE?

:CALCulate{1-16}[:SELected]:MATH:MEMorize

Description: The command stores active trace data to memory.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:CALC1:MATH:MEM**

5-29 :CALCulate{1-16}[:SELected]:MDATA Subsystem

The :CALCulate{1-16}[:SELected]:MDATA subsystem provides configuration and control for trace memory data.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:MDATA:FDATa <block>
:CALCulate{1-16}[:SELected]:MDATA:FDATa?

Description: Inputs formatted trace memory data for the active trace. The query outputs formatted trace data of the active trace.

Cmd Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10. The <block> data must exist.

Query Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10. The <block> data must exist.

Range: NA

Default Value: NA

Syntax Example: **:CALC:MDATA:FDAT <block>**

:CALC:MDATA:FDAT?

:CALCulate{1-16}[:SELected]:MDATA:SDATa <block>
:CALCulate{1-16}[:SELected]:MDATA:SDATa?

Description: Inputs S-parameter trace memory data for the active trace. The query outputs S-parameter trace memory data of the active trace.

Cmd Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:MDATA:SDAT <block>**

:CALC:MDATA:SDAT?

5-30 :CALCulate{1-16}[:SELected]:SMOothing Subsystem

The :CALCulate{1-16}[:SELected]:SMOothing subsystem commands are used to configure and control trace smoothing functions.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:SMOothing:APERture <NRf>
:CALCulate{1-16}[:SELected]:SMOothing:APERture?

Description: Sets the smoothing aperture for the indicated active trace. The query outputs the smoothing aperture for the indicated active trace.

Cmd Parameters: <NRf> The input parameter is in Percent.

Query Parameters: <NR3> The output parameter is in Percent.

Range: 0 to 100

Default Value: 0.000000E+000

Syntax Example:
```
:CALC:SMO:APER 2
:CALC:SMO:APER?
```

:CALCulate{1-16}[:SELected]:SMOothing[:STATe] <char>
:CALCulate{1-16}[:SELected]:SMOothing[:STATe]?

Description: Toggles smoothing on/off for the indicated active trace. The query outputs the smoothing on/off status for the indicated active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:
```
:CALC:SMO ON
:CALC:SMO?
```

5-31 :CALCulate{1-16}[:SELected]:TDATA Subsystem

The :CALCulate{1-16}[:SELected]:TDATA subsystem commands are used to input and report on trace data files.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

I/O Configuration and File Operation Subsystems

Related subsystems for I/O configuration and file operation are:

- ":CALCulate{1-16}:FORMat Subsystem - SnP Data" on page 5-29
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":FORMat Subsystem" on page 5-122
- ":HCOPy Subsystem" on page 5-125
- ":MMEMory Subsystem" on page 5-130

:CALCulate{1-16}[:SELected]:TDATA:FDATa <block>
:CALCulate{1-16}[:SELected]:TDATA:FDATa?

Description: Inputs formatted trace data to display on the active trace. The query outputs formatted trace data of the active trace.

Cmd Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:TDATA:FDAT <block>**

:CALC:TDATA:FDAT?

:CALCulate{1-16}[:SELected]:TDATA:SDATa <block>
:CALCulate{1-16}[:SELected]:TDATA:SDATa?

Description: Inputs S-parameter trace data to display on the active trace. The query outputs S-parameter trace data of the active trace.

Cmd Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: <block> data formatted as XML. See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:TDATA:SDAT <block>**

:CALC:TDATA:SDAT?

5-32 :CALCulate{1-16}[:SELected]:TRANsform:TIME Subsystem

The :CALCulate{1-16}[:SELected]:TRANSform:TIME subsystem commands are used to configure and control time domain transformation parameters, time domain displays, gate parameters, and window parameters. Required the Time Domain option to be installed and available.

Time Domain, Group Delay, and Reference Plane Subsystems

Related time domain, group delay, and reference place subsystems are:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":CALCulate{1-16}[:SELected]:TRANsform:TIME Subsystem" on page 5-98
- ":SENSe{1-16}:CORRection:EXTension Subsystem" on page 5-206

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

:CALCulate{1-16}[:SELected]:TRANsform:TIME:ALIAsfree?

Description: Query only. The query outputs the alias free range of the time domain transform on the active trace.

- AFT = Alias free time
- ST = The frequency step size
- AFT = 0.5 / ST immediately after reset.

Query Parameters: <NR3> The output parameter is in Seconds.

Range: –1E-9 to 4E-9

Default Value: The default for ALIAsfree depends on the ShockLine instrument model number and the installed frequency option.

Syntax Example: **:CALC:TRAN:TIME:ALIA?**

:CALCulate{1-16}[:SELected]:TRANsform:TIME:CENTer <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:CENTer?

Description: Sets the center time/distance of the range of the time domain transform on the active trace. Outputs the center time/distance of the range of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: –999.999 to 999.999 Seconds

–2.99649E11 to 21.99649E11 Meters

Default Value: 1.50000000000E-009

Syntax Example: **:CALC:TRAN:TIME:CENT 5E2**

:CALC:TRAN:TIME:CENT?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm?

Description: Sets the DC term of the time domain transform on the active trace.

- Use the AUTO value to allow the VNA instrument to determine the appropriate DC Term value.
- Use the OTHER value to allow for a user-defined DC Term value using the command:

`:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm:OTHer <NRf>`

The query returns the state of the current DC Term type.

Cmd Parameters: <char> AUTO | OTHER

Query Parameters: <char> AUTO | OTHER

Range: NA

Default Value: AUTO

Syntax Example: **:CALC:TRAN:TIME:DCT AUTO**

:CALC:TRAN:TIME:DCT?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm:OTHer <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm:OTHer?

Description: Enters the other value for the DC term of the time domain transform on the active trace. Outputs the Other value for the DC Term of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 0 to 5E3

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:TRAN:TIME:DCT:OTH 5.0E1**

:CALC:TRAN:TIME:DCT:OTH?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:DISTance?

Description: Query only. Outputs the list of time domain distance values on the active trace.

Query Parameters: <block> or <arbitrary block> See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:TRAN:TIME:DIST?**

:CALCulate{1-16}[:SELected]:TRANsform:TIME:EXTrapolate <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:EXTrapolate?

Description: Sets the extrapolation method of the time domain transform on the active trace. Outputs the extrapolation method of the time domain transform on the active trace.

Cmd Parameters: <char> MAGPHase | PHASE | USER

Query Parameters: <char> MAGPH | PHASE | USER

Range: NA

Default Value: PHASE

Syntax Example: **:CALC:TRAN:TIME:EXT MAGPH**

:CALC:TRAN:TIME:EXT?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:CENTer <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:CENTer?

Description: Sets the center time/distance of the gate of the time domain transform on the active trace. Outputs the center time/distance of the gate of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds
-2.99649E11 to 2.99649E11 Meters

Default Value: 1.50000000000E-009

Syntax Example: **:CALC:TRAN:TIME:GATE:CENT 1.0E-3**

:CALC:TRAN:TIME:GATE:CENT?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:DCGamma <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:DCGamma?

Description: Sets the Dolph-Chebyshev gamma value of the time domain transform gate on the active trace. Outputs the Dolph-Chebyshev gamma value of the time domain transform gate on the active trace.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: 0 to 2E2

Default Value: 4.00000000000E+001

Syntax Example: **:CALC:TRAN:TIME:GATE:DCG 3**

:CALC:TRAN:TIME:GATE:DCG?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:KBBeta <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:KBBeta?

Description: Sets the Kaiser-Bessel beta value of the time domain transform gate on the active trace. Outputs the Kaiser-Bessel beta value of the time domain transform gate on the active trace.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: 0 to E308

Default Value: 5.00000000000E-001

Syntax Example: **:CALC:TRAN:TIME:GATE:KBB 3**

:CALC:TRAN:TIME:GATE:KBB?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:NOTCh[:STATe] <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:NOTCh[:STATe]?

Description: Turns anti-gating on/off in the time domain transform on the active trace. Outputs the anti-gating on/off status in the time domain transform on the active trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:CALC:TRAN:TIME:GATE:NOT ON**

:CALC:TRAN:TIME:GATE:NOT?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SHAPe <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SHAPe?

Description: Sets the gate shape of the time domain transform on the active trace. The various gate shapes provide different point weighting with resultant changes in the display and its resolution. The available gate shapes are:

- DCHebyshev = Sets a Dolph-Chebyshev window with a trade-off between the side lobe level and resolution. In the DCH, the side lobe level is parameterized in absolute dB where a larger value leads to a wider main lobe width with lower resolution.
- KBEssel = Sets a Kaiser-Bessel window with a trade-off between the side lobe level and resolution. The KBE larger Beta value provides lower side lobes with a wider main lobe width with lower resolution.
- MINimum = This is a rectangular (or null) gate. It will produce the best resolution but the worst side lobe levels.
- NOMinal = The default value which provides about one-half of the resolution with no window but approximately a 30 dB reduction in side lobe levels. This setting advised for mode applications. The NOMinal setting is equivalent to a Hamming gate.
- WIDE = The WIDE gate (Blackman 3 term) will reduce resolution relative to the nominal gate but further reduce side lobe levels.
- MAXimum = The MAXimum gate (Blackman-Harris 4 term) has the poorest resolution of the fixed gates but has the lowest side lobe levels.

Outputs the gate shape of the time domain transform on the active trace.

Cmd Parameters: <char> MINimum | NOMinal | WIDE | MAXimum | DCHebyshev | KBEssel

Query Parameters: <char> MIN | NOM | WIDE | MAX | DCH | KBE

Range: NA

Default Value: NOM

Syntax Example: **:CALC:TRAN:TIME:GATE:SHAP MIN**

:CALC:TRAN:TIME:GATE:SHAP?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SPAN <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SPAN?

Description: Sets the span time/distance of the gate of the time domain transform on the active trace. Outputs the span time/distance of the gate of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds

-2.99649E11 to 2.99649E11 Meters

Default Value: 1.00000000000E-009

Syntax Example: **:CALC:TRAN:TIME:GATE:SPAN 5.0E-3**

:CALC:TRAN:TIME:GATE:SPAN?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STARt <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STARt?

Description: Sets the start time/distance of the gate of the time domain transform on the active trace. Outputs the start time/distance of the gate of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds

-2.99649E11 to 2.99649E11 Meters

Default Value: 1.00000000000E-009

Syntax Example: **:CALC:TRAN:TIME:GATE:STAR 2.0E-3**

:CALC:TRAN:TIME:GATE:STAR?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STOP <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STOP?

Description: Outputs the stop time/distance of the gate of the time domain transform on the active trace. Sets the stop time/distance of the gate of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds

-2.99649E11 to 2.99649E11 Meters

Default Value: 2.00000000000E-009

Syntax Example: **:CALC:TRAN:TIME:GATE:STOP 1.0E-2**

:CALC:TRAN:TIME:GATE:STOP?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE[:STATe] <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE[:STATe]?

Description: Sets the status of the gate of the time domain transform on the active trace, where:

- ON = The selected gate is turned on and the display adjusted appropriately.
- OFF = The gate is turned off, and the display returns to its prior state.
- DISPLAY = The upper and lower bounds of the gate are shown along with the trace display in its normal state. Use this setting to help position the gate accurately on the trace.

Cmd Parameters: <char> ON | OFF | DISPlay

Query Parameters: <char> ON | OFF | DISP

Range: NA

Default Value: OFF

Syntax Example: **:CALC:TRAN:TIME:GATE ON**

:CALC:TRAN:TIME:GATE?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:IMPUlsewidth?

Description: Query only. Outputs the impulse width of the time domain transform on the active trace.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: NA

Default Value: 0.00000000000E+000

Syntax Example: **:CALC:TRAN:TIME:IMPU?**

:CALCulate{1-16}[:SELected]:TRANsform:TIME:RESPonse <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:RESPonse?

Description: Sets the response of the time domain transform on the active trace. Outputs the response of the time domain transform on the active trace.

Cmd Parameters: <char> IMPulse | STEP

Query Parameters: <char> IMP | STEP

Range: NA

Default Value: IMP

Syntax Example: **:CALC:TRAN:TIME:RESP STEP**

:CALC:TRAN:TIME:RESP?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:SPAN <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:SPAN?

Description: Sets the span time/distance of the range of the time domain transform on the active trace. Outputs the span time/distance of the range of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds
-2.99649E11 to 2.99649E11 Meters

Default Value: 5.00000000000E-009

Syntax Example: **:CALC:TRAN:TIME:SPAN 5.0E-3**

:CALC:TRAN:TIME:SPAN?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:STARt <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:STARt?

Description: Sets the start time/distance of the range of the time domain transform on the active trace. Outputs the start time/distance of the range of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds
-2.99649E11 to 2.99649E11 Meters

Default Value: -1.00000000000E-009

Syntax Example: **:CALC:TRAN:TIME:STAR 2.0E-3**

:CALC:TRAN:TIME:STAR?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:STOP <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:STOP?

Description: Sets the stop time/distance of the range of the time domain transform on the active trace. Outputs the stop time/distance of the range of the time domain transform on the active trace.

Cmd Parameters: <NRf> The input parameter is in Seconds or Meters.

Query Parameters: <NR3> The output parameter is in Seconds or Meters.

Range: -999.99 to 999.99 Seconds
-2.99649E11 to 2.99649E11 Meters

Default Value: 4.00000000000E-009

Syntax Example: **:CALC:TRAN:TIME:STOP 1.0E-2**

:CALC:TRAN:TIME:STOP?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:TIME?

Description: Query only. Outputs the list of time domain time values on the active trace.

Query Parameters: See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:CALC:TRAN:TIME:TIME?**

:CALCulate{1-16}[:SELected]:TRANsform:TIME:TRIP <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TRIP?

Description: Sets the trip length of the time domain transform on the active trace. Outputs the trip length of the time domain transform on the active trace.

Cmd Parameters: <char> ONEway | ROUNDtrip | AUTO

Query Parameters: <char> ONE | ROUND | AUTO

Range: NA

Default Value: AUTO

Syntax Example: **:CALC:TRAN:TIME:TRIP ONE**

:CALC:TRAN:TIME:TRIP?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:TYPe <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TYPe?

Description: Sets the time domain transform type on the active trace. Outputs the time domain transform type on the active trace.

Cmd Parameters: <char> FREQuency | FREQGATE | LOWpass | BANDpass

Query Parameters: <char> FREQ | FREQGATE | LOW | BAND

Range: NA

Default Value: FREQ

Syntax Example: **:CALC:TRAN:TIME:TYP FREQ**

:CALC:TRAN:TIME:TYP?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:UNIt <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:UNIt?

Description: Sets the unit used in the time domain transform on the active trace. Outputs the unit used in time domain transform on the active trace.

Cmd Parameters: <char> TIME | DISTance

Query Parameters: <char> TIME | DIST

Range: NA

Default Value: TIME

Syntax Example: **:CALC:TRAN:TIME:UNI DIST**

:CALC:TRAN:TIME:UNI?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:DCGamma <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:DCGamma?

Description: Sets the Dolph-Chebyshev gamma value of the time domain transform window on the active trace. Outputs the Dolph-Chebyshev gamma value of the time domain transform window on the active trace.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: 0 to 2E2

Default Value: 4.00000000000E+001

Syntax Example: **:CALC:TRAN:TIME:WIND:DCG 3**

:CALC:TRAN:TIME:WIND:DCG?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:KBBeta <NRf>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:KBBeta?

Description: Sets the Kaiser-Bessel beta value of the time domain transform window on the active trace. Outputs the Kaiser-Bessel beta value of the time domain transform window on the active trace.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: 0 to E308

Default Value: 5.00000000000E-001

Syntax Example: **:CALC:TRAN:TIME:WIND:KBB 3**

:CALC:TRAN:TIME:WIND:KBB?

:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:SHAPe <char>
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:SHAPe?

Description: Sets the time window shape of the time domain transform on the active trace. The various time window shapes provide different point weighting with resultant changes in the display and its resolution. The available time window shapes are:

- RECTangular
- NOMinal
- LOWsidelobe
- MINsidelobe
- DCHebyshev
- KBEssel

Outputs the window shape of the time domain transform on the active trace.

Cmd Parameters: <char> RECTangular | NOMinal | LOWsidelobe | MINsidelobe | DCHebyshev | KBEssel

Query Parameters: <char> RECT | NOM | LOW | MIN | DCH | KBE

Range: NA

Default Value: NOM

Syntax Example: **:CALC:TRAN:TIME:WIND:SHAP RECT**

:CALC:TRAN:TIME:WIND:SHAP?

5-33 :DISPlay Subsystem

The :DISplay subsystem commands are used to control the VNA graphic display information on a per-instrument and per-trace basis.

Trace Subsystems

Related trace subsystems are:

- ":CALCulate{1-16}:PARameter and :PARameter{1-16}" on page 5-44
- ":CALCulate{1-16}:PARameter{1-16}:SELect Subsystem" on page 5-55
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:FORmat Subsystem" on page 5-67
- ":CALCulate{1-16}[:SELected]:MDATA Subsystem" on page 5-94
- ":CALCulate{1-16}[:SELected]:SMOothing Subsystem" on page 5-95
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":DISPlay Subsystem" on page 5-107

Marker Subsystems

Related marker configuration, control, and reporting subsystems are:

- ":CALCulate{1-16}:DISPlay:MARKer Subsystem" on page 5-14
- ":CALCulate{1-16}:MARKer Subsystem" on page 5-43
- ":CALCulate{1-16}:PARameter{1-16}:MARKer Subsystem" on page 5-50
- ":CALCulate{1-16}:PARameter{1-16}:MSTatistics Subsystem" on page 5-53
- ":CALCulate{1-16}[:SELected]:MARKer Subsystem" on page 5-73
- ":CALCulate{1-16}[:SELected]:MARKer{1-13} Subsystem" on page 5-88
- ":DISPlay Subsystem" on page 5-107

Limit Line Subsystems

Related limit line subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107

`:DISPlay:ANNotation:FREQuency <char>`
`:DISPlay:ANNotation:FREQuency <char>`

:

Description: Resets all displayed Marker and Marker Table frequency values to "X" if passed an "OFF" or "0". User will not be able to re-enable the display of the frequency except through Preset, recalling a setup, or restarting the application.

Cmd Parameters: <char> OFF | 0

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 1

Syntax Example: **`:DISP:ANN:FREQ?`**

:DISPlay:COLor:INVert:BACK <NRf>, <NRf>, <NRf>
:DISPlay:COLor:INVert:BACK?

Description: Sets the inverted color of the background. The invert color should be on before using this command.

Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the inverted RGB color of the background.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 255,255,255

Syntax Example: **`:DISP:COL:INV:BACK 0, 0, 0`**

`:DISP:COL:INV:BACK?`

:DISPlay:COLor:INVert:GRATicule:MAIN <NRf>, <NRf>, <NRf>
:DISPlay:COLor:INVert:GRATicule:MAIN?

Description: Sets the inverted color of the main graticule.

Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the inverted RGB color of the main graticule.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 0,0,0

Syntax Example: **`:DISP:COL:INV:GRAT:MAIN 255, 0, 0`**

`:DISP:COL:INV:GRAT:MAIN?`

:DISPlay:COLor:INVert:GRATicule:SUB <NRf>, <NRf>, <NRf>
:DISPlay:COLor:INVert:GRATicule:SUB?

Description: Sets the inverted color of the subgraticule.

Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the inverted RGB color of the subgraticule.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 100, 100, 100

Syntax Example: **:DISP:COL:INV:GRAT:SUB 255, 255, 0**

:DISP:COL:INV:GRAT:SUB?

:DISPlay:COLor:INVert:LIMit <NRf>, <NRf>, <NRf>
:DISPlay:COLor:INVert:LIMit?

Description: Sets the inverted color of the limit lines.

Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the inverted RGB color of the limit lines.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 255,0,0

Syntax Example: **:DISP:COL:INV:LIM 0, 255, 255**

:DISP:COL:INV:LIM?

:DISPlay:COLor:INVert:TRACe{1-16}:DATA <NRf>, <NRf>, <NRf>
:DISPlay:COLor:INVert:TRACe{1-16}:DATA?

Description: Sets the inverted color of the indicated data trace.

Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs inverted RGB color of the indicated data trace.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 255, 255, 0

Syntax Example: **:DISP:COL:INV:TRAC1:DATA 255, 0, 255**

:DISP:COL:INV:TRAC1:DATA?

:DISPlay:COLor:INVert:TRACe{1-16}:MEMory <NRf>, <NRf>, <NRf>
:DISPlay:COLor:INVert:TRACe{1-16}:MEMory?

Description: Sets the inverted color of the indicated memory trace. Memory trace must be on. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs inverted RGB color of the indicated memory trace.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 175, 143, 47

Syntax Example: **`:DISP:COL:INV:TRAC1:MEM 0, 0, 255`**

`:DISP:COL:INV:TRAC1:MEM?`

:DISPlay:COLor:INVert[:STATe] <char>
:DISPlay:COLor:INVert[:STATe]?

Description: Sets screen object colors to their inverted/normal color state. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the inverted/normal color state of screen objects.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **`:DISP:COL:INV ON`**

`:DISP:COL:INV?`

:DISPlay:COLor:NORMal:BACK <NRf>, <NRf>, <NRf>
:DISPlay:COLor:NORMal:BACK?

Description: Sets the normal color of the background. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the normal RGB color of the background.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 0, 0, 0

Syntax Example: **`:DISP:COL:NORM:BACK 255, 0, 255`**

`:DISP:COL:NORM:BACK?`

:DISPlay:COLor:NORMal:GRATicule:MAIN <NRf>, <NRf>, <NRf>
:DISPlay:COLor:NORMal:GRATicule:MAIN?

Description: Sets the normal color of the main graticule. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the normal RGB color of the main graticule.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 255, 255, 255

Syntax Example: **:DISP:COL:NORM:GRAT:MAIN 255, 255, 255**

:DISP:COL:NORM:GRAT:MAIN?

:DISPlay:COLor:NORMal:GRATicule:SUB <NRf>, <NRf>, <NRf>
:DISPlay:COLor:NORMal:GRATicule:SUB?

Description: Sets the normal color of the sub graticule. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the normal RGB color of the sub graticule.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 100, 100, 100

Syntax Example: **:DISP:COL:NORM:GRAT:SUB 45, 45, 45**

:DISP:COL:NORM:GRAT:SUB?

:DISPlay:COLor:NORMal:LIMit <NRf>, <NRf>, <NRf>
:DISPlay:COLor:NORMal:LIMit?

Description: Sets the normal color of the limit lines. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs the normal RGB color of the limit lines.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 255, 0, 0

Syntax Example: **:DISP:COL:NORM:LIM 100, 0, 0**

:DISP:COL:NORM:LIM?

:DISPlay:COLor:NORMal:TRACe{1-16}:DATA <NRf>, <NRf>, <NRf>
:DISPlay:COLor:NORMal:TRACe{1-16}:DATA?

Description: Sets the normal color of the indicated data trace. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs normal RGB color of the indicated data trace.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 255, 255, 0

Syntax Example: **`:DISP:COL:NORM:TRAC1:DATA 100, 100, 0`**

`:DISP:COL:NORM:TRAC1:DATA?`

`:DISPlay:COLor:NORMal:TRACe{1-16}:MEMory <NRf>, <NRf>, <NRf>`
`:DISPlay:COLor:NORMal:TRACe{1-16}:MEMory?`

Description: Sets the normal color of the indicated memory trace. Use the command below to reset the display to the factory default:

- `:DISPlay:COLor:RESet`

Outputs normal RGB color of the indicated memory trace.

Cmd Parameters: <NRf> The input parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Query Parameters: <NR1>, <NR1>, <NR1> The output parameters are integers between 0 and 255 representing the Red, Green, and Blue color values.

Range: 0 to 255

Default Value: 175, 143, 47

Syntax Example: **`:DISP:COL:NORM:TRAC1:MEM 100, 0, 100`**

`:DISP:COL:NORM:TRAC1:MEM?`

`:DISPlay:COLor:RESet`

Description: Resets all colors and inverted colors to their normal default values. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:DISP:COL:RES`**

:DISPlay:COUNt <NRf>
:DISPlay:COUNt?

Description: Sets the number of displayed channels. Command value is restricted to only (one of) the following: 1, 2, 3, 4, 6, 8, 10, 12, or 16 channels. If a number of greater than 16 is entered, the instrument is set to 16 channels. Outputs the number of displayed channels.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 16 channels.

Default Value: 1

Syntax Example: **:DISP:COUN 4**

:DISP:COUN?

:DISPlay:FSIGn[:STATe] <char>
:DISPlay:FSIGn[:STATe]?

Description: Turns on/off indicating a limit failure with a failure sign on the VNA screen. Outputs the on/off status of indicating a limit failure with a failure sign on the VNA screen.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:DISP:FSIG ON**

:DISP:FSIG?

:DISPlay:SIZe <char>
:DISPlay:SIZe?

Description: Sets the maximum/normal size of the graphic display. Outputs the maximum/normal size of the graphic display.

Cmd Parameters: <char> MAXimum | NORMal

Query Parameters: <char> MAX | NORM

Range: NA

Default: NA

Syntax Example: **:DISP:SIZ**

:DISP:SIZ?

:DISPlay:RECONFig:Clear

Description: Command is for the MS46121A/B multiple channels configuration. Reconfiguration resets the values on the active channel.

Range: N/A

Default: N/A

Syntax Example: **:DISP:RECONF:CLE**

:DISPlay:RECONFig:COUNt? <External Serial Number>

Description: Reconfiguration is a tool that allows the MS46121A/B to have multiple channels and configure them. This command is to query the number of channels open for a given MS46121A/B configuration using the reconfiguration tool.

Range: N/A

Default: N/A

Syntax Example: **:DISP:RECONF:COUN? 1234**

:DISPlay:RECONFig:STart

Description: Command is for the MS46121A/B multiple channels configuration. Reconfiguration start process.

Range: N/A

Default: N/A

Syntax Example: **:DISP:RECONF:ST**

:DISPlay:WINDow:ACTivate?

Description: Query only. Outputs the Active Channel number. To specify an active channel, use the following command:

- `:DISPlay:WINDow{1-16}:ACTivate`

Query Parameters: <NR1>

Syntax Example: **:DISP:WIND:ACT?**

:DISPlay:WINDow{1-16}:ACTivate

Description: The command sets the active channel to the indicated number. The number after WINdow is the channel activated.

This command does not support query. To query about the active channel, use the command:

- `:DISPlay:WINDow:ACTivate?`

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:DISP:WIND1:ACT**

:DISPlay:WINDow{1-16}:Y:NDIVisions <NRf>
:DISPlay:WINDow{1-16}:Y:NDIVisions?

Description: Enters the number of vertical divisions in the rectilinear displays. Outputs the number of vertical divisions in the rectilinear displays. The number of divisions must be an even number and will set an adjacent even number when other values are entered. See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 4 to 30

Default Value: 10

Syntax Example: **:DISP:WIND:Y:NDIV 4**

:DISP:WIND:Y:NDIV?

:DISPlay:WINDow{1-16}:SPLit <char>
:DISPlay:WINDow{1-16}:SPLit?

Description: Sets the trace display layout in a Row-by-Column format where R represents rows and C represents columns. If more traces are set than the trace display contains, the higher numbered trace display windows are blank. If the trace display layout is less than the number of traces set, some traces will have overlapped displays. The following trace display window arrangements are available.

- R1C1 = One trace on one row and one column
- R1C2 = Two traces, two across
- R2C1 = Two traces, two down
- R1C3 = Three traces, three across
- R3C1 = Three traces, three down
- R2C2C1 = Three traces, two on top, one on bottom
- R2C1C2 = Three traces, one on top, two on bottom
- C2R2R1 = Three traces, two on left, one on right
- C2R1R2 = Three traces, one on left, two on right
- R1C4 = Four traces, four across
- R4C1 = Four traces, four down
- R2C2 = Four traces, two across, two down
- R2C3 = Six traces, three across, two down
- R3C2 = Six traces, three down, two across
- R2C4 = Eight traces, four across, two down
- R4C2 = Eight traces, two across, four down
- R3C3 = Nine traces, three across, three down
- R5C2 = 10 traces, two across, five down
- R2C5 = 10 traces, five across, two down
- R4C3 = 12 traces, four across, three down
- R3C4 = 12 traces, three across, four down
- R4C4 = 16 traces, four across, four down

The query outputs the trace display layout.

Cmd Parameters: <char> R1C1 | R1C2 | R2C1 | R1C3 | R3C1 | R2C2C1 | R2C1C2 | C2R2R1 | C2R1R2 | R1C4 | R4C1 | R2C2 | R2C3 | R3C2 | R2C4 | R4C2 | R3C3 | R5C2 | R2C5 | R4C3 | R3C4 | R4C4

Query Parameters: <char> R1C1 | R1C2 | R2C1 | R1C3 | R3C1 | R2C2C1 | R2C1C2 | C2R2R1 | C2R1R2 | R1C4 | R4C1 | R2C2 | R2C3 | R3C2 | R2C4 | R4C2 | R3C3 | R5C2 | R2C5 | R4C3 | R3C4 | R4C4

Range: NA

Default Value: R2C2

Syntax Example: **:DISP:WIND:SPL R1C1**

:DISP:WIND:SPL?

:DISPlay:WINDow{1-16}:TITLe <string>
:DISPlay:WINDow{1-16}:TITLe?

Description: Sets the user title. Outputs the user title.

Cmd Parameters: <string>

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:DISP:WIND:TITL "Title String"**

:DISP:WIND:TITL?

:DISPlay:WINDow{1-16}:TITLe:STATe <char>
:DISPlay:WINDow{1-16}:TITLe:STATe?

Description: Enables/disables the display of the user title. Outputs the enable/disable status of the user title display.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:DISP:WIND:TITL:STAT 1**

:DISP:WIND:TITL:STAT?

:DISPlay:WINDow{1-16}:TRACe{1-16}:SIZe <char>
:DISPlay:WINDow{1-16}:TRACe{1-16}:SIZe?

Description: Sets the maximum/normal size of the indicated trace. Outputs the maximum/normal size of the indicated trace.

Cmd Parameters: <char> MAXimum | NORMal

Query Parameters: <char> MAX | NORM

Range: NA

Default Value: NORMal

Syntax Example: **:DISP:WIND:TRAC1:SIZ MAX**

:DISP:WIND:TRAC1:SIZ?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:AUTO

Description: Auto scales the display of the indicated trace. No query.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: `:DISP:WIND:TRAC1:Y:AUTO`

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV?

Description: The command enters the per-division scale value of the top display of the indicated channel and trace subject to the limitations described below. The query outputs the per-division scale value of the top display of the indicated channel and trace. See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Working with Dual-Display Rectangular Trace Formats

This command is primarily designed to be used to set the per-division (PDIV) scale value on one of the VectorStar VNA dual-display trace formats. There are four dual-display trace formats:

- Linear Magnitude and Phase (LINPHase)
- Log Magnitude and Phase (LOGPHase)
- Real and Imaginary (REIMaginary)
- Impedance Real and Imaginary (ZCOMPlex)

Working with Single-Display Rectangular or Polar Trace Formats

If the trace is a single-trace non-Smith Chart display, this command can also set the per-division scale value on single trace displays including rectangular and polar graph trace formats.

Working with Smith Chart Trace Displays

If the trace type is a Smith Impedance Chart or a Smith Admittance Chart, the available scale values are limited to the values of +3dB, 0dB, -10dB, -20dB, -30dB. These are the only values permitted and other entered values result in an execution error.

Cmd Parameters: For Smith Charts: <NRf> 3 | 0 | -10 | -20 | -30

Where:

- 3 = +3dB compressed Smith Chart
- 0 = 0 dB standard Smith Chart
- -10 = -10 dB expanded Smith Chart
- -20 = -20 dB expanded Smith Chart
- -30 = -30 dB expanded Smith Chart

For all other displays: <NRf>

Query Parameters: <NR3> The output parameter units varies depending on the trace display type.

For Smith Charts, the output parameter is based on the outside radius of the outer circle. For all other trace displays, the output is in one of the following:

- dB per division
- Hertz per division
- Meters per division
- Seconds per division.

Range: The range varies depending on display type:

- For Log Magnitude display types: 1E-3 to 1E3
- For all other non-Smith display types: 1E-5 to 1E9
- For all Smith Chart display types, discrete values only: 3 | 0 | -10 | -20 | -30.

Default Value: 1.000000E+001

Syntax Example: `:DISP:WIND1:TRAC1:Y:PDIV 1E2`

`:DISP:WIND1:TRAC1:Y:PDIV?`

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV2 <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV2?

Description: Enters the per-division scale value of the bottom display of the indicated trace. It does not create an error if the command is invoked using a one-trace display, but, when a two-trace dis lay is re-invoked, all changes made to the “hidden” trace are discarded and the trace reverts to its prior visible settings. The query outputs the per-division scale value of the bottom display of the indicated trace.

See Table 2-7, “Trace Parameters and Coefficients” on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: 1E-5 to 1E9

Default Value: 1.000000E+001

Syntax Example: **`:DISP:WIND:TRAC1:Y:PDIV2 5E0`**

`:DISP:WIND:TRAC1:Y:PDIV2?`

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:OFFSet <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:OFFSet?

Description: Enters the phase offset value for the display of the indicated trace. Outputs the phase offset value for the display of the indicated trace.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter is in Degrees.

Query Parameters: <NR3> The output parameter is in Degrees.

Range: -3.6E2 to 3.6E2

Default Value: 0.000000E+000

Syntax Example: **:DISP:WIND:TRAC1:Y:PHAS:OFFS 4.5E1**

:DISP:WIND:TRAC1:Y:PHAS:OFFS?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:WRAPping[:STATe] <char>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:WRAPping[:STATe]?

Description: Turns phase wrapping on/off on the indicated trace. This is only used with rectangular graph trace displays. The query outputs the on/off status of Phase Wrapping on the indicated trace.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:DISP:WIND:TRAC1:Y:PHAS:WRAP 1**

:DISP:WIND:TRAC1:Y:PHAS:WRAP?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHOFF <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHOFF?

Description: Enters the phase offset value for the display on the indicated trace. This is only used with rectangular graph trace displays. The query outputs the phase offset value for the display on the indicated trace.

Cmd Parameters: <NRf> The input parameter is in Degrees.

Query Parameters: <NR3> The output parameter is in Degrees.

Range: -3.6E2 to 3.6E2

Default Value: 0.00000000000E+000

Syntax Example: **:DISP:WIND:TRAC1:Y:PHOFF 2.10E1**

:DISP:WIND:TRAC1:Y:PHOFF?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV?

Description: Enters the reference level of the top display of the indicated trace. Outputs the reference level of the top display of the indicated trace.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: -9.999E2 to +9.999E2

Default Value: 0.00

Syntax Example: **:DISP:WIND:TRAC1:Y:RLEV -6.0E1**

:DISP:WIND:TRAC1:Y:RLEV?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV2 <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV2?

Description: For dual-trace displays only. Enters the reference level of the bottom display of the indicated trace. Outputs the reference level of the bottom display of the indicated trace.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: -9.999E2 to +9.999E2

Default Value: 0.00

Syntax Example: **:DISP:WIND:TRAC1:Y:RLEV2 -6.0E1**

:DISP:WIND:TRAC1:Y:RLEV2?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS?

Description: Enters the reference line position of the top display of the indicated trace. The command also works with single-display traces. Outputs the reference line position of the top display of the indicated trace.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: Minimum Reference Line Position = 0

Maximum Reference Line Position = Maximum Number of Divisions set by :DISPlay:Y:NDIVisions <NRf>

Default Value: 5

Syntax Example: **:DISP:WIND:TRAC1:Y:RPOS 10**

:DISP:WIND:TRAC1:Y:RPOS?

:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS2 <NRf>
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS2?

Description: For dual-trace displays only. Enters the reference line position of the bottom display of the indicated trace. If the trace display is a single-trace display, this command results in an execution error. Outputs the reference line position of the bottom display of the indicated trace.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: <NRf> The input parameter depends on the display type.

Query Parameters: <NR3> The output parameter depends on the display type.

Range: Minimum Reference Line Position = 0

Maximum Reference Line Position = Maximum Number of Divisions set by :DISPlay:Y:NDIVisions <NRf>

Default Value: 5

Syntax Example: **:DISP:WIND:TRAC1:Y:RPOS2 10**

:DISP:WIND:TRAC1:Y:RPOS2?

Related Cmds: **":DISPlay:Y:NDIVisions" on page 5-121**

:DISPlay:WINDow{1-16}:Y:AUTO

Description: Auto scales all traces. No query.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:DISP:WIND:Y:AUTO**

:DISPlay:Y:AUTO

Description: Auto scales all traces. No query.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:DISP:Y:AUTO**

:DISPlay:Y:NDIVisions

Description: Programs the number of vertical divisions into all rectilinear displays. No query.

See Table 2-7, "Trace Parameters and Coefficients" on page 2-19 for a complete listing of trace graph types, default settings, and available ranges.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:DISP:Y:NDIV**

5-34 :FORMat Subsystem

The :FORMat subsystem commands are used on a per-instrument basis to configure, control, and query the format for I/O data.

I/O Configuration and File Operation Subsystems

Related subsystems for I/O configuration and file operation are:

- ":CALCulate{1-16}:FORMat Subsystem - SnP Data" on page 5-29
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":FORMat Subsystem" on page 5-122
- ":HCOPy Subsystem" on page 5-125
- ":MMEMory Subsystem" on page 5-130

:FORMat:BORDer <char>
:FORMat:BORDer?

Description: Sets the binary numeric I/O data byte order as either Most Significant Byte (MSB) or Least Significant Byte (LSB) byte order of binary numeric I/O data where:

- NORMal = The MSB is first
- SWAPped = The LSB is first.

The query outputs the Most Significant Byte (MSB)/Least Significant Byte (LSB) byte order of binary numeric I/O data.

Cmd Parameters: <char> NORMal | SWAPped

Query Parameters: <char> NORM | SWAP

Range: NA

Default Value: SWAP

Syntax Example: **:FORM:BORD NORM**

:FORM:BORD?

:FORMat:DATA <char>
:FORMat:DATA?

Description: Sets the format for numeric I/O data representation where:

- ASCii = An ASCII number of 20 or 21 characters long with floating point notation. For example, 12345E-4.
- REAL = 8 Bytes of binary floating point number representation limited to 64 bits which is the value of the MPND (Maximum Positive/Negative Double Precision Number or +/- 1.792 631 348 6E38)
- REAL32 = 4 Bytes of floating point number representation which is the value of the MPNF (Maximum Positive/Negative Floating Point Number or +/- 3.402 819E38)

The query outputs the format of numeric I/O data representation.

Cmd Parameters: <char> ASCii | REAL | REAL32

Query Parameters: <char> ASC | REAL | REAL32

Range: NA

Default Value: ASC

Syntax Example: **:FORM:DATA REAL**

:FORM:DATA?

:FORMat:DATA:HEADing[:STATe] <char>
:FORMat:DATA:HEADing[:STATe]?

Description: Enables or disables including a heading with data files. Outputs the enable/disable status of including a heading with data files.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 1

Syntax Example: **:FORM:DATA:HEAD ON**

:FORM:DATA:HEAD?

:FORMat:SNP:FREQuency <char>
:FORMat:SNP:FREQuency?

Description: Sets the frequency unit displayed in an SNP data file. Queries the frequency unit displayed in an SNP data file.

Cmd Parameters: <char> HZ | KHZ | MHZ | GHZ

Query Parameters: <char> HZ | KHZ | MHZ | GHZ

Range: NA

Default Value: GHZ

Syntax Example: **:FORM:SNP:FREQ HZ**

:FORM:SNP:FREQ?

:FORMat:SNP:PARameter <char>
:FORMat:SNP:PARameter?

Description: Sets the parameter format displayed in an SNP data file. Outputs the parameter format displayed in an SNP data file.

Cmd Parameters: <char> LINPH | LOGPH | REIM

Where:

- LINPH = Linear and Phase
- LOGPH = Log and Phase

- REIM = Real and Imaginary Numbers

Query Parameters: <char> LINPH | LOGPH | REIM

Range: NA

Default Value: REIM

Syntax Example: **:FORM:SNP:PAR LINPH**

:FORM:SNP:PAR?

5-35 :HCOPy Subsystem

The :HCOPy subsystem commands are used to create print output default settings for the instrument graphics display.

I/O Configuration and File Operation Subsystems

Related subsystems for I/O configuration and file operation are:

- ":CALCulate{1-16}:FORMat Subsystem - SnP Data" on page 5-29
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":FORMat Subsystem" on page 5-122
- ":HCOPy Subsystem" on page 5-125
- ":MMEMory Subsystem" on page 5-130

:HCOPy[:IMMediate]

Description: Prints the display image to the default printer. The default printer is set through Windows print setup.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:HCOP**

:HCOPy:DEVice:ID <String>
:HCOPy:DEVice:ID?

Description: : The command enters the Device Identity string for the header printout. The query outputs the Device Identify string for the header printout.

Query Parameters: NA

Cmd Parameters: <String>

Output: <char>

Range: NA

Default: NA

Syntax Example: :HCOP:DEV:ID <char>

:HCOP:DEV:ID?

:HCOPy:DEVice:ID:STATe <char>
:HCOPy:DEVice:ID:STATe?

Description: The command enters the on/off state of the Device Identity string for the header printout. The query outputs the on/off state of the Device Identify string for the header printout.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: NA

Output: : <char> 1 | 0

Range: NA

Default: NA

Syntax Example: :HCOP:DEV:ID:STAT <char>

:HCOP:DEV:ID:STAT?

:HCOPy:IMAGe <char>
:HCOPy:IMAGe?

Description: Sets the hardcopy print color for the display where:

- NORMal = As the instrument screen is displayed.
- INVert = Inverts the colors. Typically used to print the traces as colors, and the instrument display background as white.
- BWHITe = Converts all colors to either black or white.

Outputs the hardcopy print color for the display.

Cmd Parameters: <char> NORMal | INVert | BWHITe

Default Value: INV

Syntax Example: :HCOP:IMAG NORM

:HCOPy:IMAGe?

:HCOPy:MODel <String>
:HCOPy:MODel?

Description: Enter the Model string for the header printout. Output the Model string for the header printout

Cmd Parameters: NA

Query Parameters: NA

Output: <char>

Range: NA

Default: NA

Syntax Example: :HCOP:MOD <String>

:HCOP:MOD?

:HCOPy:MODel:STATe<char>
:HCOPy:MODel:STATe?

Description: Enter the on/off state of the Model string for the header printout. Output the on/off state of the Model string for the header printout.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default: NA

Syntax Example: :HCOP:MOD:STAT <char>

:HCOP:MOD:STAT?

:HCOPy:OPERator:COMMent<String>
:HCOPy:OPERator:COMMent?

Description: Enter the Operator Comment string for the header printout. Output the Operator Command string for the header printout.

Cmd Parameters: <String>

Query Parameters: NA

Output: <char>

Range: NA

Default: NA

Syntax Example: :HCOP:OPER:COMM <String>

:HCOP:OPER:COMM?

:HCOPy:OPERator:COMMent:STATe <char>
:HCOPy:OPERator:COMMent:STATe?

Description: Enters the on/off state of Enter the Operator Comment string for the header printout. Output the on/off state of the Operator Command string for the header printout.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: NA

Output: <char> 1 | 0

Range: NA

Default: NA

Syntax Example: :HCOP:OPER:COMM:STAT <char>

:HCOP:OPER:COMM:STAT?

:HCOPy:OPERator:NAMe <String>
:HCOPy:OPERator:NAMe?

Description: Enter the Operator Name string for the header printout. Output the Operator Name string for the header printout.

Cmd Parameters: <String>

Query Parameters: NA

Output: <char>

Range: NA

Default: NA

Syntax Example: :HCOP:OPER:NAM <String>

:HCOP:OPER:NAM?

:HCOPy:OPERator:NAMe:STATe <char>
:HCOPy:OPERator:NAMe:STATe?

Description: Enter the on/off state of the Operator Name string for the header printout. Output the on/off state of the Operator Name string for the header printout.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: NA

Output: <char> 1 | 0

Range: NA

Default: NA

Syntax Example: :HCOP:OPER:NAM:STAT <char>

:HCOP:OPER:NAM:STAT?

:HCOPy:PRINt:DATe:TIMe:STATe <char>
:HCOPy:PRINt:DATe:TIMe:STATe?

Description: Turns on/off printing the Date Time information. Output the on/off state of printing the Date Time information.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: NA

Output: <char> 1 | 0

Range: NA

Default: NA

Syntax Example: :HCOP:PRIN:DAT:TIM:STAT <char>

:HCOP:PRIN:DAT:TIM:STAT?

:HCOPy:PRINt:HEADers:STATe <char>
:HCOPy:PRINt:HEADers:STATe?

Description: Turns on/off printing the header information. Output the on/off state of printing the header information.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: NA

Output: <char> 1|0

Range: NA

Default: NA

Syntax Example: :HCOP:PRIN:HEAD:STAT <char>

:HCOP:PRIN:HEAD:STAT?

:HCOPy:PRINt:LOGO:STATe <char>
:HCOPy:PRINt:LOGO:STATe?

Description: Turns on/off printing a Logo bitmap at the top of the print page. Output the on/off state of printing a Logo bitmap at the top of the print page.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: NA

Output: <char> 1|0

Range: NA

Default: NA

Syntax Example: :HCOP:PRIN:LOGO:STAT <char>

:HCOP:PRIN:LOGO:STAT?

:HCOPy:PRINt:LOGO:TYPe <char>
:HCOPy:PRINt:LOGO:TYPe?

Description: Sets the logo type to print at the top of the print page. Queries the logo type to print at the top of the print page.

Cmd Parameters: <char> ANRitsu|USER

Where:
- ANRitsu = standard Anritsu corporate logo
- USER = user defined logo

Query Parameters: NA

Output: <char> ANR | USER

Range: NA

Default: NA

Syntax Example: :HCOP:PRIN:LOGO:TYP <char>

:HCOP:PRIN:LOGO:TYP?

:HCOPy:PRINt:TYPe <char>
:HCOPy:PRINt:TYPe?

Description: Select the hardcopy print type for the HCOPy print com

5-36 :MMEMory Subsystem

The :MMEMory subsystem commands are used to input, load, and read out instrument data files. <String> formatted data is generally used to represent file directories and file names.

I/O Configuration and File Operation Subsystems

Related subsystems for I/O configuration and file operation are:

- ":CALCulate{1-16}:FORMat Subsystem - SnP Data" on page 5-29
- ":CALCulate{1-16}[:SELected]:DATA Subsystem" on page 5-65
- ":CALCulate{1-16}[:SELected]:TDATA Subsystem" on page 5-96
- ":FORMat Subsystem" on page 5-122
- ":HCOPy Subsystem" on page 5-125
- ":MMEMory Subsystem" on page 5-130

:MMEMory:CATalog? {<string>}

Description: Query only. Read out the directory information of the storage device. If the directory does not exist, it results in an execution error.

Query Parameters: <string> Path specification in the form: 'x:\directory' where x: must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: Default path is "C:\" if not specified.

Syntax Example: **:MMEM:CAT? 'C:\directory'**

:MMEMory:COPY <string1>, <string2>

Description: Copies the contents of the first <string1> file to the second <string2> file. The directory and file for <string1> must exist. If the directory and file for <string2> do not exist, they will be created. If the directory and file for <string1> already exist, they will be overwritten. No query.

Cmd Parameters: <string1> Filename and path in the form: 'x:\directory\filename.xxx' where x:\directory\filename.xxx must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:COPY 'C:\filename1.s2p', 'C:\filename2.s2p'**

:MMEMory:DELete <string>

Description: Deletes a disk, file, or directory. Use caution with this command as there is no recovery operation in case of a user mistake or error. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.xxx' or 'x:\directory' where x:\directory\filename.xxx or x:\directory must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:DEL 'C:\directory\filename.s2p'**

:MMEMory:LOAD <string>

Description: Loads data file whose type is specified by the filename extension into the instrument memory. The directory and file in <string> must exist. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.xxx' where x:\directory\filename.xxx must exist.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:LOAD 'C:\filename.s2p'**

:MMEMory:LOAD:CKIT <string>

Description: Loads a calibration kit file or multiple files from the given file specification into the instrument memory. The :MMEMory:LOAD:CKIT command supports loading XML files with a .ccf extension or binary files. No query.

Cmd Parameters: <string> Filename and path where the required format changes depending on the file type being loaded.

If a .ccf file type is used, the required form is: 'x:\directory\filename.ccf' where x:\directory\filename.ccf must exist.

If a binary file type is used, only the path is required in the form: 'x:\directory' where x:\directory and cal kit files must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: For calibration kit or other .ccf files:

```
:MMEM:LOAD:CKIT 'E:\calibrationfoldermykit.ccf'
```

For binary files:

```
:MMEM:LOAD:CKIT 'E:\calibrationbinaries'
```

:MMEMory:LOAD:FSEGMent <string>

Description: Loads the frequency-based segmented sweep table from the given filespec. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.sgs' where x:\directory\filename.sgs must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:LOAD:FSEGM 'C:\directory\filename.sgs'**

:MMEMory:LOAD:ISEGMent <string>

Description: Loads the index-based segmented sweep table from the given file specification. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.sgs' where x:\directory\filename.sgs must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:LOAD:ISEGM 'C:\directory\filename.sgs'**

:MMEMory:LOAD:LIMit <string>

Description: Loads a Limit Table from the file system. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.lmt' where x:\directory\filename.lmt must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:LOAD:LIM 'C:\limits3.lmt'**

:MMEMory:LOAD:MDATA <string>

Description: Loads formatted or unformatted trace data from a file into the trace memory of the active trace. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.tdf' or 'x:\directory\filename.tdu' where x:\directory\filename.tdf or x:\directory\filename.tdf must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:LOAD:MDATA 'C:\directory\filename.tdu'**

:MMEMory:MDIRectory <string>

Description: Create a new disk directory. No query.

Cmd Parameters: <string> Path specification in the form: 'x:\directory' where x: must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:MDIR 'C:\directory'**

:MMEMory:RDIRectory <string>

Description: Delete a disk directory. No query.

Cmd Parameters: <string> Path specification in the form: 'x:\directory' where x:\ and \directory\ must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:RDIR 'C:\directory'**

:MMEMory:STORe <string>

Description: Stores a data file of the type specified by the filename extension.No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.xxx' where x:\directory\ must exist. See definition of "<string>" on page 2-10 and Table 2-8 for a list of supported file types.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:STOR 'C:\filename.acd'**

:MMEMory:STORe:FSEGMent <string>

Description: Stores the frequency-based segmented sweep table to the given filespec. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.sgs' where x:\directory\ must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:STOR:FSEGM 'C:\directory\filename.sgs'**

:MMEMory:STORe:IMAGe <string>

Description: Save the display image to a disk file (JPEG, PNG, or BMP format). No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.bmp' where x:\directory\ must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:STOR:IMAG 'C:\directory\filename.bmp'**

:MMEMory:STORe:ISEGMent <string>

Description: Stores the index-based segmented sweep table to the given filespec. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.sgs' where x:\directory\ must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:STOR:ISEGM 'C:\directory\filename.sgs'**

:MMEMory:STORe:LIMit <string>

Description: Stores the limit table to the hard disk.No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.lmt' where x:\directory must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:STOR:LIM 'C:\directory\filename.lmt'**

:MMEMory:STORe:MDATA <string>

Description: Save formatted or unformatted trace memory data to the hard disk from the active trace. No query.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.tdf' or 'x:\directory\filename.tdu' where x:\directory must exist. See definition of "<string>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:MMEM:STOR:MDATA 'C:\directory\filename.tdu'**

:MMEMory:TRANsfer <string>, <block>
:MMEMory:TRANsfer?

Description: Writes data to a disk file. The query outputs the disk file data. The file must exist.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.xxx' where x:\directory must exist.

<block> Binary data block must exist.

Query Parameters: <string> The filename and path in the form: 'x:\directory\filename.xxx' where x:\directory\filename.xxx must exist.

See definition of "<string>" on page 2-10 and "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: `:MMEM:TRAN 'C:\directory\filename.cal', <block>`

`:MMEM:TRAN? 'C:\directory\filename.cal'`

:MMEMory:WRITE:CKIT <char1>, <char2>, <char3>, <string>

Description: Writes calibration kit file from the current setup.

User supplies the parameters for Line Type, Cal Type, and specified connector type (Coaxial, Waveguide, or User Defined), and file name parameter string. No query is supported.

Definitions for Required Parameters <char1> <char2><char3>

<char1> is Line Type = COAXial | NONDispersive | WAVEguide.
Microstrip is not an available option.

<char2> is Cal Type = SOLX | SSLT | SSST | LRL/LRM
For SOLT calibration methods, use the SOLX parameter.

<char3> is Connector Type (Coaxial, Waveguide or User Defined).

CID1 = W1
CIDV = V
CIDK = K
CID2 = 2.4 mm
CID3 = GPC-3.5
CIDS = SMA
CIDN = N
CIDN75 = N-connector, 75 Ohm
CIDG = GPC-7
CID716 = 7/16 inch
CIDT = TNC
CID = WR-28 Waveguide
CIDWR42 = WR-42 Waveguide
CIDWR64 = WR-64 Waveguide
CIDU1 = User defined 1[a]
CIDU2 = User defined 2
CIDU3 = User defined 3

CIDU4 = User defined 4
CIDU5 = User defined 5
CIDU6 = User defined 6
CIDU7 = User defined 7
CIDU8 = User defined 8

a. Note that a user-defined CIDU1 connector can be assigned a user-defined name, but programmatically, it must be referred to as "CIDU1". This also applies to CIDU2 through CIDU8.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their parameters.

Definitions for Required Parameter <string>

The <string> parameter provides the file name, and optionally the directory name, subject to the limitations described below.

User-provided variations in the <string> parameter command suffix and existing O/S conditions change the way the command will operate.

- Best practices recommend stating a file name and path in the form 'x:\directory\filename.xxx' where 'x:\directory\' already exists.
- If the directory and file name in <string> already exist, the command will overwrite them without comment.
- If the directory portion of the <string> is missing such as 'x:\filename.xxx', the named file is placed in the current O/S defined default directory.
- If the command uses a <string> in the path and file name form such as 'x:\directory\filename.xxx' but 'x:\directory\' does not exist, an error is generated and no file is written.
- See definition of "<string>" on page 2-10.

Cmd Parameters: <char1> COAXial | NONDispersive | WAVEguide

<char2> SOLX | SSLT | SSST

<char3> <char3> CID1 | CIDV | CIDK | CID2 | CID3 | CIDS | CIDN | CIDN75 | CIDG | CID716 | CIDT | CIDWR28 | CIDWR42 | CIDWR64 | CIDU1 | CIDU2 | CDU3 | CIDU4 | CIDU5 | CIDU6 | CIDU7 | CIDU8

<string> Path and filename in the form 'x:\directory\filename.xxx' or 'x:\filename.xxx'. If the 'x:\directory\' form is used, the directory must exist. See <string> parameter definitions above and definition of "<string>" on page 2-10.

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:MMEM:WRITE:CKIT COAX, SOLX, CID3, 'C:\directory\calfilename.ccf'`**

5-37 :SENSe{1-16}:ABORtcal Subsystem

The :SENSe{1-16}:ABORtcal subsystem command aborts the calibration.

Calibration Subsystems with Actual Calibrations

Related calibration subsystems that perform actual calibrations are:

- ":SENSe{1-16}:ABORtcal Subsystem" on page 5-136
- ":SENSe{1-16}:CORRection:COLLect:PORT Subsystem" on page 5-157
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189

:SENSe{1-16}:ABORtcal

Description: Aborts the current hardware or RF calibration. No query.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS{1-16}:ABOR**

5-38 :SENSe{1-16}:AVERage Subsystem

The :SENSe{1-16}:AVERage subsystem commands configure and control the averaging function.

Sweep Subsystems

Related sweep configuration and control subsystems are:

- Section 5-38 :SENSe{1-16}:AVERage Subsystem on page 5-136
- Section 5-55 :SENSe{1-16}:FREQuency Subsystem on page 5-209
- Section 5-63 :SENSe{1-16}:SWEep Subsystem on page 5-236

:SENSe{1-16}:AVERage:CLEar

Description: Clears and restarts the averaging sweep count. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:AVER:CLE**

:SENSe{1-16}:AVERage:COUNt <NRf>
:SENSe{1-16}:AVERage:COUNt?

Description: Sets the averaging count. Outputs the averaging count.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 1024

Default Value: 1

Syntax Example: **:SENS:AVER:COUN 101**

:SENS:AVER:COUN?

:SENSe{1-16}:AVERage:SWEep?

Description: Query only. Outputs the averaging sweep count.

Query Parameters: <NR1> The output parameter is an integer.

Range: NA

Default Value: 0

Syntax Example: **:SENS:AVER:SWE?**

:SENSe{1-16}:AVERage:TYPe <char>
:SENSe{1-16}:AVERage:TYPe?

Description: Sets the averaging function type to point-by-point or sweep-by-sweep. Outputs the averaging function type of point-by-point or sweep-by-sweep.

Cmd Parameters: <char> POINtbypoint | SWEepbysweep

Query Parameters: <char> POIN | SWE

Range: NA

Default Value: POIN

Syntax Example: **:SENS:AVER:TYP POIN**

:SENS:AVER:TYP?

:SENSe{1-16}:AVERage[:STATe] <char>
:SENSe{1-16}:AVERage[:STATe]?

Description: Turns averaging on/off. Outputs the averaging function on/off status.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:SENS:AVER ON**

:SENS:AVER?

5-39 :SENSe{1-16}:BANDwidth Subsystem

The :SENSe{1-16}:BANDwidth subsystem command sets the IF bandwidth. Note that this command is the same as the :SENSe{1-16}:BWIDth command:

IF Configuration Subsystems

Related IF configuration and control subsystems are:

- Section 5-39 :SENSe{1-16}:BANDwidth Subsystem on page 5-138
- Section 5-40 :SENSe{1-16}:BWIDth Subsystem on page 5-139

:SENSe{1-16}:BANDwidth[:RESolution] <NRf>
:SENSe{1-16}:BANDwidth[:RESolution]?

Description: Sets the IF bandwidth. The query outputs the IF bandwidth.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: 1 to 5E5. The system will automatically select the closest IF bandwidth from the following options:

- 10, 30, 50, 70, 100, 300, 500, 700 Hz
- 1, 3, 5, 7, 10, 30, 50, 70, 100, 300, 500 kHz

Default Value: 1.00000000000E+003

Syntax Example: **:SENS:BAND 4**

:SENS:BAND?

Related Cmds: **:SENSe{1-16}:BWIDth[:RESolution] <NRf>**

5-40 :SENSe{1-16}:BWIDth Subsystem

The :SENSe{1-16}:BWIDth subsystem command sets the IF bandwidth. Note that this command is the same as the :SENSe{1-16}:BANDwidth command.

IF Configuration Subsystems

Related IF configuration and control subsystems are:

- Section 5-39 :SENSe{1-16}:BANDwidth Subsystem on page 5-138
- Section 5-40 :SENSe{1-16}:BWIDth Subsystem on page 5-139

:SENSe{1-16}:BWIDth[:RESolution] <NRf>
:SENSe{1-16}:BWIDth[:RESolution]?

Description: Sets the IF bandwidth. The query outputs the IF bandwidth. A related command is:

- :SENSe{1-16}:BANDwidth[:RESolution] <NRf> on page 5-138

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: 1 to 5E5. The system will automatically select the closest IF bandwidth from the following options:

- 10, 30, 50, 70, 100, 300, 500, 700 Hz
- 1, 3, 5, 7, 10, 30, 50, 70, 100, 300, 500 kHz

Default Value: 1.00000000000E+003

Syntax Example: **:SENS:BWID 3100**

:SENS:BWID?

5-41 :SENSe{1-16}:CORRection:COEFficient:PORT

The :SENSe{1-16}:CORRection:COEFficient:PORT subsystem commands are used to simulate a instrument calibration on a 2-Port VNA instrument so that a set of user-defined calibration parameters can be input using the :SENSe{1-16}:CORRection:COEFficient <char>,<block> command. Note that this subsystem does not perform an actual calibration.

Calibration Simulation Subsystems

These subsystems are used to create a calibrated state in the instrument which is followed by adding the required error correction coefficients for the required calibration type. If this approach is used, each error correction coefficient is entered by separate commands. Simulated calibration subsystems are:

- Section 5-41 :SENSe{1-16}:CORRection:COEFficient:PORT on page 5-140
- Section 5-42 :SENSe{1-16}:CORRection:COEFficient Subsystem on page 5-144

Calibration Type Abbreviations

The calibration abbreviations and their calibration types are:

- :1P2PF refers to a one-path two-port calibration forward direction
- :1P2PR refers to a one path two port calibration reverse direction
- :FULL1 refers to a full one port calibration
- :FULL2 refers to a full two port calibration
- :FULLB refers to a full one port reflection calibration on both ports
- :RESP1 refers to a one port response calibration
- :RESPB refers to a one port response calibration both ports
- :TFRB refers to a transmission frequency response calibration both directions
- :TFRF refers to a transmission frequency response calibration forward direction
- :TFRR refers to a transmission frequency response calibration reverse direction

Each calibration simulation type command is described in greater detail in the individual command descriptions below.

Related Query

These commands set the Calibration Type. To query which Calibration Type has been set, use the command:

```
:SENSe{1-16}:CORRection:COLLect:TYPe?
```

:SENSe{1-16}:CORRection:COEFficient <char>, <block>
:SENSe{1-16}:CORRection:COEFficient? <char>

Description: Inputs the values for a single calibration correction coefficient such as ED1. The command identifies the coefficient name and then the coefficient data for a calibration. Separate commands are issued for each required correction coefficient. The query outputs the values of a correction coefficient of the calibration.

Cmd Parameters: <char> ED1 | EP1S | ET11 | ET21 | EP2L | EX21 | ED2 | EP2S | ET22 | ET12 | EP1L | EX12

<block> The second input parameter is a block ASCII value. The actual ASCII block must be constructed and sent.

Query Parameters: <char> ED1 | EP1S | ET11 | ET21 | EP2L | EX21 | ED2 | EP2S | ET22 | ET12 | EP1L | EX12

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF ED1, <block>**

:SENS:CORR:COEF? ED1

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:1P2PF

Description: Simulates a One-Path Two-Port Calibration Forward Direction on the indicated port set. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

To perform an actual calibration, use:

:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:1P2PF

Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:1P2PF**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:1P2PR

Description: Simulates a One-Path Two-Port Calibration Reverse Direction on the indicated port set. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:1P2PR**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:FULL2

Description: Simulates a Full Two-Port Calibration on the indicated port set. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:FULL2**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:FULLB

Description: Simulates a Full One-Port Reflection Calibration on both ports. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:FULLB**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:RESPB

Description: Simulates a One-Port Response Calibration on both ports on the indicated port set. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:RESPB**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:TFRB

Description: Simulates a Transmission Frequency Response Calibration in both directions on the indicated port set.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:TFRB**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:TFRF

Description: Simulates a Transmission Frequency Response Calibration in the forward direction on the indicated port set. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:TFRF**

:SENSe{1-16}:CORRection:COEFficient:PORT{12}:TFRR

Description: Simulates a Transmission Frequency Response Calibration in the reverse direction on the indicated port set. No query.

To query the state of this command use:

:SENSe{1-16}:CORRection:COLLect:TYPe?

No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT12:TFRR**

:SENSe{1-16}:CORRection:COEFficient:PORT{1-2}:FULL1

Description: Simulates a Full One-Port Reflection Calibration on the indicated port. No query.

To query the state of this command use:

```
:SENSe{1-16}:CORRection:COLLect:TYPe?
```

No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT1:FULL1**

:SENSe{1-16}:CORRection:COEFficient:PORT{1-2}:RESP1

Description: Simulates a One-Port Response Calibration on the indicated port. No query.

To query the state of this command use:

```
:SENSe{1-16}:CORRection:COLLect:TYPe?
```

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COEF:PORT1:RESP1**

5-42 :SENSe{1-16}:CORRection:COEFficient Subsystem

The :SENSe{1-16}:CORRection:COEFficient subsystem commands are used to simulate a instrument calibration on 2-Port VNA instruments so that a set of user-defined calibration parameters can be input using the :SENSe{1-16}:CORRection:COEFficient <char>,<block> command. Note that this subsystem does not perform an actual calibration.

Calibration Simulation Subsystems

These subsystems are used to create a calibrated state in the instrument which is followed by adding the required error correction coefficients for the required calibration type. If this approach is used, each error correction coefficient is entered by separate commands. Simulated calibration subsystems are:

- :SENSe{1-16}:CORRection:COEFficient:PORT on page 5-140
- :SENSe{1-16}:CORRection:COEFficient Subsystem on page 5-144

Calibration Type Abbreviations

The calibration abbreviations and their calibration types are:

- :1P2PF refers to a one-path two-port calibration forward direction
- :1P2PR refers to a one path two port calibration reverse direction
- :FULL1 refers to a full one port calibration
- :FULL2 refers to a full two port calibration
- :FULLB refers to a full one port reflection calibration on both ports
- :RESP1 refers to a one port response calibration
- :RESPB refers to a one port response calibration both ports
- :TFRB refers to a transmission frequency response calibration both directions
- :TFRF refers to a transmission frequency response calibration forward direction
- :TFRR refers to a transmission frequency response calibration reverse direction

Each calibration simulation type command is described in greater detail in the individual command descriptions below.

These commands set the Calibration Type. To query which Calibration Type has been set, use the command:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

:SENSe{1-16}:CORRection:COEFficient[:METHod]:1P2PF

Description: Simulates a one-path two-port calibration forward direction. No query.

To query the state of this command use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:1P2PF
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:SENS:CORR:COEF:1P2PF`**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:1P2PR

Description: Simulates a one-path Two-Port Calibration Reverse Direction. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:1P2PR
```

No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:1P2PR**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:FULL1

Description: Simulates a full One-Port Reflection Calibration. No query.

Before sending this command, the simulation port must be specified using:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT
```

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:FULL1
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:FULL1**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:FULL2

Description: Simulates a full Two-Port Calibration. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:FULL2
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:FULL2**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:FULLB

Description: Simulates a full One-Port Reflection Calibration on both ports. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:FULLB
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:FULLB**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:RESP1

Description: Simulates a One-Port Response Calibration. No query.

Before sending this command, the simulation port must be specified using:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT
```

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:RESP1
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:RESP1**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:RESPB

Description: Simulates a One-Port Response Calibration on both ports. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:RESPB
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:RESPB**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:TFRB

Description: Simulates a Transmission Frequency Response Calibration in both directions. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRB
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COEF:TFRB**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:TFRF

Description: Simulates a Transmission Frequency Response Calibration in the forward direction. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRF
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:SENS:CORR:COEF:TFRF`**

:SENSe{1-16}:CORRection:COEFficient[:METHod]:TFRR

Description: Simulates a Transmission Frequency Response Calibration in the reverse direction. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To set this calibration mode and then perform an actual calibration, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRR
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:SENS:CORR:COEF:TFRR`**

5-43 :SENSe{1-16}:CORRection:COLLect:ECAL Subsystems

The :SENSe{1-16}:CORRection:COLLect:ECAL subsystem commands set the automatic calibration parameters.

:SENSe{1-16}:CORRection:COLLect:ECAL:AUTOmatic:ORIentation[:STATe] <char>
:SENSe{1-16}:CORRection:COLLect:ECAL:AUTOmatic:ORIentation[:STATe]?

Description: The command turns automatic Autocal module orientation detection on/off for the given channel. Query outputs the on/off status of the Autocal module orientation detection on the given channel.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default: NA

Syntax Example: SENS1:CORR:COLL:ECAL:AUTO:ORI ON

SENS1:CORR:COLL:ECAL:AUTO:ORI?

:SENSe{1-16}:CORRection:COLLect:ECAL:BEGin?

Description: The query starts an Autocal calibration and returns status or next step information.

Cmd Parameters: NA

Query Parameters: <NR1> Outputs one of the following ECAL List of Messages:

0 - AssurancePassed

1 - Update

2 - TrueThru

3 - Adapter

4 - NoModule

5 - NoOrient

6 - NoFile

7 - NoMatch

8 - No12T

9 - NotAllowed

10 - OutOfRange

11 - AssuranceFailed

12 - Aborted

13 - AbortOK

14 - AbortNotOK

15 - ACError

16 - ACFatalError

17 - DoneCalculateCoeff

18 - ACConnectCalB

19 - CharacBad

20 - DisplayMessage

21 - ConnectToPort1

22 - ConnectToPort2

23 - ConnectToPort1

24 - ConnectThrubwPorts12

25 - SequentialBegins

Range: NA

Default: NA

Syntax Example: :SENS1:CORR:COLL:ECAL:BEG?

:SENSe{1-16}:CORRection:COLLect:ECAL:CONTinue?

Description: The query starts an Autocal characterization and returns status or next step information.

Cmd Parameters: NA

Query Parameters: <NR1> Outputs one of the following ECAL List of Messages:

0 - AssurancePassed

1 - Update

2 - TrueThru

3 - Adapter

4 - NoModule

5 - NoOrient

6 - NoFile

7 - NoMatch

8 - No12T

9 - NotAllowed

10 - OutOfRange

11 - AssuranceFailed

12 - Aborted

13 - AbortOK

14 - AbortNotOK

15 - ACError

16 - ACFatalError

17 - DoneCalculateCoeff

18 - ACConnectCalB

19 - CharacBad

20 - DisplayMessage

21 - ConnectToPort1

22 - ConnectToPort2

23 - ConnectToPorts12

24 - ConnectThrubwPorts12 - SequentialBegins

Range: NA

Default: NA

Syntax Example: :SENS1:CORR:COLL:ECAL:CONT?

:SENSe{1-16}:CORRection:COLLect:ECAL:ORIentation <char>
:SENSe{1-16}:CORRection:COLLect:ECAL:ORIentation?

Description: The command turns Autocal module and SmartCal orientation detection off and sets the orientations manually for the given channel. A query outputs the Autocal module and SmartCal orientations of the given channel.

Cmd Parameters: <char> L1 | L2 | R1 | R2 | L1R2 | R1L2 | R2L1 | L2R1

<char> A1 | A2 | B1 | B2 | A1B2 | B1A2 | B2A1 |A2B1

Query Parameters: <char> L1 | L2 | R1 | R2 | L1R2 | R1L2 | R2L1 | L2R1

<char> A1 | A2 | B1 | B2 | A1B2 | B1A2 | B2A1 |A2B1

Range: NA

Default: L1R2

Syntax Example: :SENS1:CORR:COLL:ECAL:ORI L2R1

:SENS1:CORR:COLL:ECAL:ORI A2B1

:SENS1:CORR:COLL:ECAL:ORI?

:SENS1:CORR:COLL:ECAL:ORI?

:SENSe{1-16}:CORRection:COLLect:ECAL[:CALa]:TRUEthru <char>
:SENSe{1-16}:CORRection:COLLect:ECAL[:CALa]:TRUEthru?

Description: The command turns on/off the use of TrueThru during Autocal CALA Calibration on the given channel.

Cmd Parameters: <char> 1|0|ON|OFF

Query Parameters: <char> 1|0

Range: NA

Default: 0

Syntax Example: :SENS1:CORR:COLL:ECAL:TRUE ON

:SENS1:CORR:COLL:ECAL:TRUE?

5-44 :SENSe{1-16}:CORRection:COLLect:METHod Subsystem

The :SENSe{1-16}:CORRection:COLLect:METHod subsystem command sets the calibration method. When configuring the instrument calibration this option must be set first, generally followed by line type, and finally calibration type. Optionally, these commands are followed by coefficient and characterization commands for user-defined calibration devices and kits.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:COLLect:METHod <char>
:SENSe{1-16}:CORRection:COLLect:METHod?

Description: The command sets the calibration method for the indicated channel. The following calibration methods are available:

- AUTOcal = Automatic Calibrator (AutoCal) Module calibration method using Anritsu 36585-Series Precision AutoCal Modules
- SMARtcal = USB SmartCal Module calibration method using Anritsu MS25xxx Series SmartCal Modules
- SOLT = Short-Open-Load-Through calibration method
- SSLT = Short-Short-Load-Through calibration method
- SSST = Short-Short-Short-Through calibration method
- LRL = Line-Reflect-Line calibration method.
- LRM = Line Reflect-Match calibration method.

The query outputs the calibration method for the indicated channel.

Cmd Parameters: <char> AUTOcal | SMARtcal | SOLT | SSLT | SSST | LRL/LRM

Query Parameters: <char> AUTO | SMAR | SOLT | SSLT | SSST | LRL/LRM

Range: NA

Default Value: SOLT

Syntax Example: **:SENS1:CORR:COLL:METH AUTO**

:SENS1:CORR:COLL:METH?

5-45 :SENSe{1-16}:CORRection:COLLect:MICrostrip Subsystem

The :SENSe{1-16}:CORRection:COLLect:MICrostrip subsystem commands set the parameter values for dielectric, kit type, and port assigned for microstrip substrate values.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:IMPedance:TRANsformation Subsystem" on page 5-41
- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MICrostrip Subsystem" on page 5-152
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect:WAVeguide" on page 5-199
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:COLLect:MICrostrip:DIELectric <NRf>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:DIELectric?

Description: Sets the microstrip substrate dielectric value for calibration on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their command parameters. Outputs the microstrip substrate dielectric value for calibration on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Query Parameters.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:MIC:DIEL 1.2E0`

`:SENS1:CORR:COLL:MIC:DIEL?`

:SENSe{1-16}:CORRection:COLLect:MICrostrip:EFFective <NRf>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:EFFective?

Description: Sets the microstrip effective dielectric value for calibration on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters. Outputs the microstrip effective dielectric value for calibration on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:MIC:EFF 1.2E0`

`:SENS1:CORR:COLL:MIC:EFF?`

:SENSe{1-16}:CORRection:COLLect:MICrostrip:KIT <char>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:KIT?

Description: Selects the microstrip kit to use for calibration on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Available Microstrip Kits

The available microstrip kit parameters are:

- MIL10 = Standard 10-mil (0.010" or 0.25400 mm thick) microstrip
- MIL15 = Standard 15-mil (0.015" or 0.38100 mm thick) microstrip
- MIL25 = Standard 10-mil (0.025" or 0.63500 mm thick) microstrip

USERx = User-defined microstrip 1 through 32 (e.g., USER5, USER23)**Using User-Defined Microstrip Kits**

User-defined microstrips (USER1 through USER8 above) can be defined through the menu-driven user interface by entering six values:

- Microstrip Kit Label = Defaults as "User-DefinedN" (where N = 1 to 8) and can be changed as required. Programmatically, the each user-defined microstrip kit name must be still referred to as the appropriate "USERn" parameter.
- Strip Width (mm). Programmatically, the width is entered in Meters.
- Impedance (Ohms)
- Substrate Thickness (mm). Programmatically, the thickness is entered in Meters.
- Substrate Dielectric Value
- Effective Dielectric Value

Query Output

Outputs the microstrip kit selected for calibration on the indicated channel.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <char> MIL10 | MIL15 | MIL25 | USER1 | USER2 | USER3 | USER4 | USER5 | USER6 | USER7 | USER8

Query Parameters: <char> MIL10 | MIL15 | MIL25 | USER1 | USER2 | USER3 | USER4 | USER5 | USER6 | USER7 | USER8

Range: NA

Default Value: MIL10

Syntax Example: **:SENS1:CORR:COLL:MIC:KIT MIL15**

:SENS1:CORR:COLL:MIC:KIT?

:SENSe{1-16}:CORRection:COLLect:MICrostrip:PORT{1-2}:CONNector <char>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:PORT{1-2}:CONNector?

Description: Sets the microstrip kit connector type for the indicated port on the indicated channel to where only user-defined microstrips can be used in the <char> parameter as:

- USERx = User-defined microstrip 1 through 32 (e.g., USER5, USER23)

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their command parameters.

User-Defined Microstrips

In the menu-driven user interface, user-defined microstrips are set with six values:

- Microstrip Kit Label = Defaults as "User-DefinedN" (where N = 1 to 8) and can be changed as required.
- Strip Width (mm). Programmatically, the width is entered in Meters.
- Impedance (Ohms)
- Substrate Thickness (mm). Programmatically, the thickness is entered in Meters.
- Substrate Dielectric Value
- Effective Dielectric Value

The query outputs the microstrip kit connector type for the indicated port on the indicated channel. See Table 2-16, "Connector Type Abbreviations and Descriptions" on page 2-33 for a complete listing of calibration components, connectors, and their parameters.

Cmd Parameters: <char> MIL10 | MIL15 | MIL25 | USER1 | USER2 | USER3 | USER4 | USER5 | USER6 | USER7 | USER8

Query Parameters: <char> MIL10 | MIL15 | MIL25 | USER1 | USER2 | USER3 | USER4 | USER5 | USER6 | USER7 | USER8

Range: NA

Default Value: CMV

Syntax Example: **:SENS1:CORR:COLL:MIC:PORT1:CONN USER2**

:SENS1:CORR:COLL:MIC:PORT1:CONN?

:SENSe{1-16}:CORRection:COLLect:MICrostrip:THICKness <NRf>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:THICKness?

Description: Sets the microstrip substrate thickness for calibration on the indicated channel.

Outputs the microstrip substrate thickness for calibration on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:MIC:THICK 1.2E-5**

:SENS1:CORR:COLL:MIC:THICK?

:SENSe{1-16}:CORRection:COLLect:MICrostrip:WIDth <NRf>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:WIDth?

Description: Sets the microstrip width for calibration on the indicated channel.

Outputs the microstrip width for calibration on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:MIC:WID 1.2E-4**

:SENS1:CORR:COLL:MIC:WID?

:SENSe{1-16}:CORRection:COLLect:MICrostrip:Z0 <NRf>
:SENSe{1-16}:CORRection:COLLect:MICrostrip:Z0?

Description: Sets the microstrip impedance (Z zero) for calibration on the indicated channel. Outputs the microstrip impedance for calibration on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:MIC:Z0 7.5E1**

:SENS1:CORR:COLL:MIC:Z0?

5-46 :SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem

The:SENSe{1-16}:CORRection:COLLect:MULTIple subsystem command sets transmission through (thru) lines between one or more port pairs.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:COLLect:MULTiple:THRu <char> {<char>,<char>,<char>,<char>}
:SENSe{1-16}:CORRection:COLLect:MULTiple:THRu?

Description: Adds one or more Transmission Throughs (Thrus) to the calibration process. Outputs the calibration process list of Transmission Throughs.

Cmd Parameters: <char> THRu12

Query Parameters: <char> THR12

Range: NA

Default Value: NA

Syntax Example:
```
:SENS:CORR:COLL:MULT:THR THR12
:SENS:CORR:COLL:MULT:THR?
```

5-47 :SENSe{1-16}:CORRection:COLLect:PORT Subsystem

The :SENSe{1-16}:CORRection:COLLect:PORT subsystem commands start an actual instrument calibration.

Calibration Subsystems with Actual Calibrations

Related calibration subsystems that perform actual calibrations are:

- ":SENSe{1-16}:ABORtcal Subsystem" on page 5-136
- ":SENSe{1-16}:CORRection:COLLect:PORT Subsystem" on page 5-157
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189

Calibration Type Abbreviations

The commands in this subsystem use the following abbreviations for different calibration methods:

- 1P2PF = One-path two-port calibration, forward direction
- 1P2PR = One path two port calibration, reverse direction
- FULL1 = Full one port calibration
- FULL2 = Full two port calibration
- FULLB = Full one port reflection calibration, both ports
- RESP1 = One port response calibration
- RESPB = One port response calibration, both ports
- TFRB = Transmission frequency response calibration, both directions
- TFRF = Transmission frequency response calibration, forward direction
- TFRR = Transmission frequency response calibration, reverse direction

:SENSe{1-16}:CORRection:COLLect:PORT{1|2|12}:FULL1

Description: Selects Full One-Port Calibration as the calibration type on the indicated ports selected. The Full 1 port calibration can be performed on any combination of ports.No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:FULL1**

:SENSe{1-16}:CORRection:COLLect:PORT{1 | 2 | 12}:RESP1

Description: Selects One-Port Response Calibration as the calibration type on the indicated ports selected. No query.

The response calibration can be performed on one or more ports by listing them using as PORTw | wx | wxy | wxyz where:

- w = the first port
- x = the second port
- y = the third port
- z = the fourth port

For example, to perform a response calibration on Ports 1, 2 use:

```
SENS:CORR:COLL:PORT12:RESP1
```

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:RESP1**

:SENSe{1-16}:CORRection:COLLect:PORT{12}:ISOL

Description: Initiates collection of the isolation standard data for the calibration. For a 2-Port VNA, the available port pair is 12. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:ISOL**

:SENSe{1-16}:CORRection:COLLect:PORT{12}:TFRB

Description: Adds Transmission Frequency Response Calibration Both directions on the indicated port pair. The Transmission Frequency Response calibration can be any combination of port pairs, and any combination of Forward only (TFRF), Reverse only (TFRR) or both (TFRB). No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:TFRB**

:SENSe{1-16}:CORRection:COLLect:PORT{12}:TFRF

Description: Adds Transmission Frequency Response Calibration Forward direction on the indicated port pair. The Transmission Frequency Response calibration can be any combination of port pairs, and any combination of Forward only (TFRF), Reverse only (TFRR) or both (TFRB). No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:TFRF**

:SENSe{1-16}:CORRection:COLLect:PORT{12}:TFRR

Description: Adds Transmission Frequency Response Calibration Reverse direction on the indicated port pair. The Transmission Frequency Response calibration can be any combination of port pairs, and any combination of Forward only (TFRF), Reverse only (TFRR) or both (TFRB). No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:TFRR**

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu

Description: Initiates collection of the through (thru) standard data for the calibration. For 2-Port VNAs, the available port pair is 12. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:THR**

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:FREQuency <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:FREQuency?

Description: Sets the through-line (thru-line) reference frequency of the loss on the selected port-pair. Outputs the thru-line reference frequency of the loss on the selected port-pair.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT12:THR:FREQ 1.0E6**

:SENS:CORR:COLL:PORT12:THR:FREQ?

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LABEL?

Description: Sets the through-line (thru-line) label on the selected port-pair. On 3-Port VNAs, the available port pair is 12. Outputs the thru-line label on the selected port-pair.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

See definition of "<char>" on page 2-12.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:THR:LABEL 'IC7000'**

:SENS:CORR:COLL:PORT12:THR:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LENGth <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LENGth?

Description: Sets the through-line (thru-line) length on the selected port-pair. The port-pair for 2-Port VNAs is Port 1-2. Outputs the thru-line length on the selected port-pair.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT12:THR:LENG 1.0E0**

:SENS:CORR:COLL:PORT12:THR:LENG?

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LOSS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LOSS?

Description: Sets the through-line (thru-line) loss on the selected port-pair. The port-pair for 2-Port VNAs is Port 1-2. Outputs the thru-line loss on the selected port-pair.

Cmd Parameters: <NRf> The input parameter is in dB/mm.

Query Parameters: <NR3> The output parameter is in dB/mm.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT12:THR:LOSS 3.0E0**

:SENS:CORR:COLL:PORT12:THR:LOSS?

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:RECIProcal <char>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:RECIProcal?

Description: Sets the through-line (thru-line) use reciprocal flag on the selected port-pair. The port-pair for 2-Port VNAs is Port 1-2. Outputs the thru-line use reciprocal flag on the selected port-pair.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA>

Default Value: 0

Syntax Example: **:SENS:CORR:COLL:PORT12:THR:RECIP ON**

:SENS:CORR:COLL:PORT12:THR:RECIP?

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:SERial?

Description: Sets the through-line (thru-line) serial number on the selected port-pair. The port-pair for 2-Port VNAs is Port 1-2. Outputs the thru-line serial number on the selected port-pair.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT12:THR:SER '123456'**

:SENS:CORR:COLL:PORT12:THR:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:Z0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:Z0?

Description: Sets the through-line (thru-line) impedance on the selected port-pair. The port-pair for 2-Port VNAs is Port 1-2. Outputs the thru-line impedance on the selected port-pair. The port-pair for 2-Port VNAs is Port 1-2.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 1E-4 to 1E10

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS:CORR:COLL:PORT12:THR:Z0 7.5E1`

`:SENS:CORR:COLL:PORT12:THR:Z0?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:CONNector <char>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:CONNector?

Description: Sets the connector type for the indicated port. The connector types are:

- CF1 = 1 mm (female)
- CF2 = 2.4 mm (female)
- CF3 = GPC 3.5 (female)
- CF716 = 7/16 (female)
- CFK = K (female)
- CFKT = K (female) TOSLKF Cal Kit
- CFNT = N (female) TOSLNF Cal Kit
- CFN = N (female)
- CFN75 = N 75 Ohm (female)
- CFC = TNC (female)
- CFS = SMA (female)
- CFV = V (female)
- CFUn = User-defined female connector from 1 to 8, where: CFU1 = User-defined female 1, CFU2 = User-defined female 2, CFU3 = User-defined female 3, CFU4 = User-defined female 4, CFU5 = User-defined female 5, CFU6 = User-defined female 6, CFU7 = User-defined female 7, CFU8 = User-defined female 8
- CM1 = 1 mm (male)
- CM2 = 2.4 mm (male)
- CM3 = GPC 3.5 (male)
- CM716 = 7/16 (male)
- CMC = TNC (male)
- CMK = K (male)
- CMKT = K (male) TOSLK Cal Kit
- CMNT = N (male) TOSLN Cal Kit
- CMN = N (male)
- CMN75 = N 75 Ohm (male)
- CMS = SMA (male)
- CMV = V (male)
- CMUn = User-defined male connector from 1 to 8, where: CMU1 = User-defined male 1, CMU2 = User-defined male 2, CMU3 = User-defined male 3, CMU4 =

User-defined male 4, CMU5 = User-defined male 5, CMU6 = User-defined male 6, CMU7 = User-defined male 7, CMU8 = User-defined male 8.

- CNG = GPC7 (none)

Outputs the connector type for the indicated port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <char> CM1 |CM2 |CM3 | CMK | CMN | CMS |CMC |CMKT | CMNT | CMV | CM716 | CNG | CMN75 | CMU1 | CMU2 | CMU3 | CMU4 | CMU5 | CMU6 | CMU7 | CMU8 | CF2 | CFK | CFN | CFS | CFC |CFKT |CFNT | CFV | CF1 | CF716 | CFN75 | CFU1 | CFU2 | CFU3 | CFU4 | CFU5 |CFU6 | CFU7 |CFU8

Query Parameters: <char> CM1 |CM2 |CM3 | CMK | CMN | CMS |CMC |CMKT | CMNT | CMV | CM716 | CNG | CMN75 | CMU1 | CMU2 | CMU3 | CMU4 | CMU5 | CMU6 | CMU7 | CMU8 | CF2 | CFK | CFN | CFS | CFC |CFKT |CFNT | CFV | CF1 | CF716 | CFN75 | CFU1 | CFU2 | CFU3 | CFU4 | CFU5 |CFU6 | CFU7 |CFU8

Range: NA

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:CONN CMN**

:SENS:CORR:COLL:PORT1:CONN?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD

Description: Initiates collection of the Load Standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:SELect <char>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:SELect?

Description: Selects the load standard of LOAD1 or LOAD2 to be used for calibration on the indicated port. Outputs the selected load standard of LOAD1 or LOAD2 to be used for calibration on the indicated port.

Cmd Parameters: <char> LOAD1 | LOAD2

Query Parameters: <char> LOAD1 | LOAD2

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD:SEL LOAD2**

:SENS:CORR:COLL:PORT1:LOAD:SEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:TYPe <char>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:TYPe?

Description: Selects the Broadband Load Type as Fixed or Sliding for calibration on the indicated port. Outputs the Broadband Load Type selection on the indicated port.

Cmd Parameters: <char> FIXed | SLIDing

Query Parameters: <char> FIX | SLID

Range: NA

Default: FIX

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD:TYP FIX**

:SENS:CORR:COLL:PORT1:LOAD:TYP?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C0?

Description: Sets the Load1 standard C0 (C zero) capacitance on the selected port. The C0 coefficient is measured in Farads. Outputs the Load1 standard C0 capacitance on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Farads.

Cmd Parameters: <NR3> The output parameter is in Farads.

Range: 0 to 1E12

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:C0 3.0E-11**

:SENS:CORR:COLL:PORT1:LOAD1:C0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C1?

Description: Sets the Load1 Standard C1 (C one) Coefficient on the selected port. The C1 coefficient is measured in Farads/Hertz. Outputs the Load1 Standard C1 Coefficient on the Selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz.

Query Parameters: <NR3> The output parameter is in Farads/Hertz.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:C1 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:C1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C2?

Description: Sets the Load1 Standard C2 Coefficient on the selected port. The C2 coefficient is measured in Farads/Hertz^2. Outputs the Load1 Standard C2 Coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^2 (Hertz E2).

Query Parameters: <NR3> The output parameter is in Farads/Hertz^2.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:C2 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:C2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C3?

Description: Sets the Load1 Standard C3 Coefficient on the selected port. The C3 coefficient is measured in Farads/Hertz^3. Outputs the Load1 Standard C3 Coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^3 (Hertz E3).

Query Parameters: <NR3> The output parameter is in Farads/Hertz^3.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:C3 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:C3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L0?

Description: Sets the Load1 standard L0 (L zero) inductance on the selected port. The L0 coefficient is measured in Henrys. Outputs the Load1 standard L0 inductance on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:L0 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:L0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L1?

Description: Sets the Load1 standard L1 (L one) inductance coefficient on the selected port. The L1 coefficient is measured in Henrys/Hertz. Outputs the Load1 standard L1 inductance coefficient on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:L1 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:L1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L2?

Description: Sets the Load1 standard L2 inductance coefficient on the selected port. The L2 coefficient is measured in Henrys'/Hertz^2. Outputs the Load1 standard L2 inductance coefficient on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^2.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^2.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:L2 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:L2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L3?

Description: Sets the Load1 standard L3 inductance coefficient on the selected port. The L3 coefficient is measured in Henrys/Hertz^3. Outputs the Load1 standard L3 inductance coefficient on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^3.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^3.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:L3 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:L3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:LABEL?

Description: Sets the Load1 standard label on the selected port. Outputs the Load1 standard label on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:LABEL 'IC7000'**

:SENS:CORR:COLL:PORT1:LOAD1:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF1?

Description: Sets the Load1 Standard Offset length1 coefficient on the selected port. Outputs the Load1 Standard Offset length1 coefficient on the Selected port.

Cmd Parameters: <NRf> The input parameter is in Meters/Hertz.

Query Parameters: <NR3> The output parameter is in Meters/Hertz.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:OFF1 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:OFF1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF2?

Description: Sets the Load1 Standard Offset length2 coefficient on the Selected port. Outputs the Load1 Standard Offset length2 coefficient on the Selected port.

Cmd Parameters: <NRf> The input parameter is in Meters/Hertz^2.

Query Parameters: <NR3> The output parameter is in Meters/Hertz^2.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:OFF2 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:OFF2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF3?

Description: Sets the Load1 Standard Offset length3 coefficient on the Selected port. Outputs the Load1 Standard Offset length3 coefficient on the Selected port.

Cmd Parameters: <NRf> The input parameter is in Meters/Hertz^3.

Query Parameters: <NR3> The output parameter is in Meters/Hertz^3.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:OFF3 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD1:OFF3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFFS?

Description: Sets the Load1 (Load one) standard offset on the selected port. Outputs the Load1 standard offset on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:OFFS 2.5E1**

:SENS:CORR:COLL:PORT1:LOAD1:OFFS?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:R <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:R?

Description: Sets the Load1 (Load one) standard resistance on the selected port. Outputs the Load1 standard resistance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 0 to 1E10

Default Value: 5.00000000000E+001

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:R 7.5E1**

:SENS:CORR:COLL:PORT1:LOAD1:R?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:SERial?

Description: Sets the Load1 (Load one) standard serial number on the selected port. Outputs the Load1 standard serial number on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:SER '123456'**

:SENS:CORR:COLL:PORT1:LOAD1:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:Z0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:Z0?

Description: Sets the Load1 (Load one) standard Impedance on the selected port. Outputs the Load1 standard Impedance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 1E-4 to 1E10

Default Value: 5.00000000000E+001

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD1:Z0 7.5E1**

:SENS:CORR:COLL:PORT1:LOAD1:Z0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C0?

Description: Sets the Load2 standard C0 (C zero) capacitance on the selected port. The C0 coefficient is measured in Farads. Outputs the Load2 standard C0 capacitance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads.

Query Parameters: <NR3> The output parameter is in Farads.

Range: 0 to 1E12

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:C0 2.0E-11**

:SENS:CORR:COLL:PORT1:LOAD2:C0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C1?

Description: Sets the Load2 Standard C1 (C one) Coefficient on the Selected port. The C1 coefficient is measured in Farads/Hertz. Outputs the Load2 Standard C1 Coefficient on the Selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz.

Query Parameters: <NR3> The output parameter is in Farads/Hertz.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:C1 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:C1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C2?

Description: Sets the Load2 Standard C2 Coefficient on the selected port. The C2 coefficient is measured in Farads/Hertz^2. Outputs the Load2 Standard C2 Coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^2.

Query Parameters: <NR3> The output parameter is in Farads/Hertz^2.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:C2 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:C2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C3?

Description: Sets the Load2 Standard C3 Coefficient on the selected port. The C3 coefficient is measured in Farads/Hertz^3. Outputs the Load2 Standard C3 Coefficient on the Selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^3.

Query Parameters: <NR3> The output parameter is in Farads/Hertz^3.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:C3 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:C3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L0?

Description: Sets the Load2 standard L0 (L zero) inductance on the selected port. The L0 coefficient is measured in Henrys. Outputs the Load2 standard L0 inductance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:L0 2.0E-6**

:SENS:CORR:COLL:PORT1:LOAD2:L0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L1?

Description: Sets the Load2 standard L1 (L one) inductance coefficient on the selected port. The L1 coefficient is measured in Henrys/Hertz. Outputs the Load2 standard L1 inductance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:L1 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:L1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L2?

Description: Sets the Load2 standard L2 inductance coefficient on the selected port. The L2 coefficient is measured in Henrys/Hertz^2. Outputs the Load2 standard L2 inductance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^2.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^2.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:L2 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:L2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L3?

Description: Sets the Load2 standard L3 inductance coefficient on the selected port. The L3 coefficient is measured in Henrys/Hertz^3. Outputs the Load2 standard L3 inductance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^3.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^3.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:L3 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:L3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:LABEL?

Description: Sets the Load2 standard label on the selected port. Outputs the Load2 standard label on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:LABEL 'IC7000'**

:SENS:CORR:COLL:PORT1:LOAD2:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF1?

Description: Sets the Load2 Standard Offset length1 (length one) coefficient on the selected port. The OFF1 coefficient is measured in Meters/Hertz. Outputs the Load2 Standard Offset length1 coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Meters/Hertz.

Query Parameters: <NR3> The output parameter is in Meters/Hertz.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:OFF1 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:OFF1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF2?

Description: Sets the Load2 Standard Offset length2 coefficient on the selected port. The OFF2 coefficient is measured in Meters/Hertz^2. Outputs the Load2 Standard Offset length2 coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Meters/Hertz^2.

Query Parameters: <NR3> The output parameter is in Meters/Hertz^2.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:OFF2 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:OFF2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF3?

Description: Sets the Load2 Standard Offset length3 coefficient on the selected port. The OFF3 coefficient is measured in Meters/Hertz^3. Outputs the Load2 Standard Offset length3 coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Meters/Hertz^3.

Query Parameters: <NR3> The output parameter is in Meters/Hertz^3.

Range: MPND

Default: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:OFF3 2.0E0**

:SENS:CORR:COLL:PORT1:LOAD2:OFF3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFFS?

Description: Sets the Load2 standard offset on the selected port. Outputs the Load2 standard offset on the selected port.

Query Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:OFFS 2.5E1**

:SENS:CORR:COLL:PORT1:LOAD2:OFFS?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:R <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:R?

Description: Sets the Load2 standard resistance on the selected port. Outputs the Load2 standard resistance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 0 to 1E10

Default Value: 5.00000000000E+001

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:R 7.5E1**

:SENS:CORR:COLL:PORT1:LOAD2:R?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:SERial?

Description: Sets the Load2 standard serial number on the selected port. Outputs the Load2 standard serial number on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:SER '123456'**

:SENS:CORR:COLL:PORT1:LOAD2:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:Z0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:Z0?

Description: Sets the Load2 standard Impedance on the selected port. Outputs the Load2 standard Impedance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 1E-4 to 1E10

Default Value: 5.00000000000E+001

Syntax Example: **:SENS:CORR:COLL:PORT1:LOAD2:Z0 7.5E1**

:SENS:CORR:COLL:PORT1:LOAD2:Z0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN

Description: Initiates collection of the open standard data for the calibration on the given port.

No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:OPEN**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C0?

Description: Sets the open standard C0 (C zero) capacitance on the selected port. The C0 coefficient is measured in Farads. Outputs the open standard C0 capacitance on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads.

Query Parameters: <NR3> The output parameter is in Farads.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:OPEN:C0 2.0E-11**

:SENS:CORR:COLL:PORT1:OPEN:C0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C1 <NRF>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C1?

Description: Sets the open standard C1 (C one) capacitance coefficient on the selected port. The C1 coefficient is measured in Farads/Hertz. Outputs the open standard C1 capacitance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz.

Query Parameters: <NR3> The output parameter is in Farads/Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS:CORR:COLL:PORT1:OPEN:C1 2.0E0`

`:SENS:CORR:COLL:PORT1:OPEN:C1?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C2?

Description: Sets the open standard C2 capacitance coefficient on the selected port. The C2 coefficient is measured in Farads/Hertz^2. Outputs the open standard C2 capacitance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^2.

Query Parameters: <NR3> The output parameter is in Farads/Hertz^2.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS:CORR:COLL:PORT1:OPEN:C2 2.0E0`

`:SENS:CORR:COLL:PORT1:OPEN:C2?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C3?

Description: Sets the open standard C3 capacitance coefficient on the selected port. The C3 coefficient is measured in Farads/Hertz^3. Outputs the open standard C3 capacitance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^3.

Query Parameters: <NR3> The output parameter is in Farads/Hertz^3.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS:CORR:COLL:PORT1:OPEN:C3 2.0E0`

`:SENS:CORR:COLL:PORT1:OPEN:C3?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:LABEL?

Description: Sets the open standard label on the selected port. Outputs the open standard label on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:OPEN:LABEL 'IC7000'**

:SENS:CORR:COLL:PORT1:OPEN:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:OFFS?

Description: Sets the open standard offset on the selected port. The OFFS parameter is measured in Meters. Outputs the open standard offset on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:OPEN:OFFS 2.5E1**

:SENS:CORR:COLL:PORT1:OPEN:OFFS?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:SERial?

Description: Sets the open standard serial number on the selected port. Outputs the open standard serial number on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:OPEN:SER '123456'**

:SENS:CORR:COLL:PORT1:OPEN:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:REFLection:COMPonent <char>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:REFLection:COMPonent?

Description: Selects the reflection standard to use for the reflection calibration on the indicated port where the reflection standard types can be:

- OPEN = Open
- SHORt = Short
- NONE = Only returned by the query if not reflection component has been defined.

Outputs the reflect standard to use for the reflection calibration on the indicated port.

Cmd Parameters: <char> OPEN | SHORt

Query Parameters: <char> OPEN | SHOR | NONE

Range: NA

Default: OPEN

Syntax Example: **:SENS:CORR:COLL:PORT1:REFL:COMP OPEN**

:SENS:CORR:COLL:PORT1:REFL:COMP?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT

Description: Initiates collection of the short standard data for the calibration on the given port. No query.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their command parameters.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L0?

Description: Sets the short standard L0 (L zero) inductance on the selected port. The L0 coefficient is measured in Henrys. Outputs the short standard L0 inductance on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:L0 2.0E0**

:SENS:CORR:COLL:PORT1:SHORT:L0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L1?

Description: Sets the short standard L1 (L one) inductance coefficient on the selected port. The L1 coefficient is measured in Henrys/Hertz. Outputs the short standard L1 inductance coefficient on the selected port.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their command parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:L1 2.0E0**

:SENS:CORR:COLL:PORT1:SHORT:L1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L2?

Description: Sets the short standard L2 inductance coefficient on the selected port. The L2 coefficient is measured in Henrys/Hertz^2. Outputs the short standard L2 inductance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^2.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^2.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:L2 2.0E0**

:SENS:CORR:COLL:PORT1:SHORT:L2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L3?

Description: Sets the short standard L3 inductance coefficient on the selected port. The L3 coefficient is measured in Henrys/Hertz^3. Outputs the short standard L3 inductance coefficient on the selected port.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^3.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^3.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:L3 2.0E0**

:SENS:CORR:COLL:PORT1:SHORT:L3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:LABEL?

Description: Sets the short standard label on the selected port. Outputs the short standard label on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:LABEL 'IC7000'**

:SENS:CORR:COLL:PORT1:SHORT:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:OFFS?

Description: Sets the short standard offset on the selected port. Outputs the short standard offset on the selected port.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:OFFS 2.5E1**

:SENS:CORR:COLL:PORT1:SHORT:OFFS?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:SERial?

Description: Sets the short standard serial number on the selected port. Outputs the short standard serial number on the selected port.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SHORT:SER '123456'**

:SENS:CORR:COLL:PORT1:SHORT:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1

Description: Initiates collection of the Offset Short1 (Short one) standard data for the calibration on the given channel and port. See "Calibration Component Parameters" on page 2-31. for a complete listing of calibration components, connectors, and their Command Parameters.

No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L0?

Description: Sets the Short1 standard L0 (L zero) inductance on the selected port of the indicated channel. The L0 parameter is measured in Henrys. Outputs the Short1 standard L0 inductance on the selected port of the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:L0 2.0E0**

:SENS1:CORR:COLL:PORT1:SHORT1:L0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L1?

Description: Sets the Short1 standard L1 (L one) inductance coefficient on the selected port of the indicated channel. The L1 coefficient is measured in Henrys/Hertz. Outputs the Short1 standard L1 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:L1 2.0E0**

:SENS1:CORR:COLL:PORT1:SHORT1:L1?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L2?

Description: Sets the Short1 standard L2 inductance coefficient on the selected port of the indicated channel. The L2 coefficient is measured in Henrys/Hertz^2. Outputs the Short1 standard L2 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^2.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^2.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:L2 2.0E0**

:SENS1:CORR:COLL:PORT1:SHORT1:L2?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L3?

Description: Sets the Short1 standard L3 inductance coefficient on the selected port of the indicated channel. The L3 coefficient is measured in Henrys/Hertz^3. Outputs the Short1 standard L3 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^3.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^3.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:L3 2.0E0**

:SENS1:CORR:COLL:PORT1:SHORT1:L3?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:LABEL?

Description: Sets the Short1 (Short one) standard label on the selected port of the indicated channel. Outputs the Short1 standard label on the selected port of the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:LABEL 'IC7000'**

:SENS1:CORR:COLL:PORT1:SHORT1:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:OFFS?

Description: Sets the Short1 (Short 1) standard offset on the selected port of the indicated channel. Outputs the Short1 standard offset on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:OFFS 2.5E1**

:SENS1:CORR:COLL:PORT1:SHORT1:OFFS?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:SERial?

Description: Sets the Short1 (Short 1) standard serial number on the selected port of the indicated channel. Outputs the Short1 standard serial number on the selected port of the indicated channel.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT1:SER '123456'**

:SENS1:CORR:COLL:PORT1:SHORT1:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2

Description: Initiates collection of the Offset Short2 standard data for the calibration on the given channel and port.

No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT2**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L0?

Description: Sets the Short2 standard L0 (L zero) inductance on the selected port of the indicated channel. The L0 (zero) coefficient is measured in Henrys. Outputs the Short2 standard L0 inductance on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT2:L0 2.0E0**

:SENS1:CORR:COLL:PORT1:SHORT2:L0?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L1?

Description: Sets the Short2 standard L1 (L one) inductance coefficient on the selected port of the indicated channel. The L1 coefficient is measured in Henrys/Hertz. Outputs the Short2 standard L1 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:PORT1:SHORT2:L1 2.0E0`

`:SENS1:CORR:COLL:PORT1:SHORT2:L1?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L2?

Description: Sets the Short2 standard L2 inductance coefficient on the selected port of the indicated channel. The L2 coefficient is measured in Henrys/Hertz^2. Outputs the Short2 standard L2 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^2.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^2.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:PORT1:SHORT2:L2 2.0E0`

`:SENS1:CORR:COLL:PORT1:SHORT2:L2?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L3?

Description: Sets the Short2 standard L3 inductance coefficient on the selected port of the indicated channel. The L3 coefficient is measured in Henrys/Hertz^3. Outputs the Short2 standard L3 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^3.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^3.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:PORT1:SHORT2:L3 2.0E0`

`:SENS1:CORR:COLL:PORT1:SHORT2:L3?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:LABEL?

Description: Sets the Short2 standard label on the selected port of the indicated channel. Outputs the Short2 standard label on the selected port of the indicated channel.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT2:LABEL 'IC7000'**

:SENS1:CORR:COLL:PORT1:SHORT2:LABEL?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:OFFS?

Description: Sets the Short2 standard offset on the selected port of the indicated channel. Outputs the Short2 standard offset on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT2:OFFS 2.5E1**

:SENS1:CORR:COLL:PORT1:SHORT2:OFFS?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:SERial?

Description: Sets the Short2 standard serial number on the selected port of the indicated channel. Outputs the Short2 standard serial number on the selected port of the indicated channel.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT2:SER '123456'**

:SENS1:CORR:COLL:PORT1:SHORT2:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3

Description: Initiates collection of the Offset Short3 standard data for the calibration on the given channel and port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT3**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L0?

Description: Sets the Short3 standard L0 (L zero) inductance on the selected port of the indicated channel. The L0 coefficient is measured in Henrys. Outputs the Short3 standard L0 inductance on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Query Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:PORT1:SHORT3:L0 2.0E0`

`:SENS1:CORR:COLL:PORT1:SHORT3:L0?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L1 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L1?

Description: Sets the Short3 standard L1 (L one) inductance coefficient on the selected port of the indicated channel. The L1 coefficient is measured in Henrys/Hertz. Outputs the Short3 standard L1 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:PORT1:SHORT3:L1 2.0E0`

`:SENS1:CORR:COLL:PORT1:SHORT3:L1?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L2 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L2?

Description: Sets the Short3 standard L2 inductance coefficient on the selected port of the indicated channel. The L2 coefficient is measured in Henrys/Hertz^2. Outputs the Short3 standard L2 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^2.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^2.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:PORT1:SHORT3:L2 2.0E0`

`:SENS1:CORR:COLL:PORT1:SHORT3:L2?`

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L3 <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L3?

Description: Sets the Short3 standard L3 inductance coefficient on the selected port of the indicated channel. The L3 coefficient is measured in Henrys/Hertz^3. Outputs the Short3 standard L3 inductance coefficient on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Henrys/Hertz^3.

Query Parameters: <NR3> The output parameter is in Henrys/Hertz^3.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example:
```
:SENS1:CORR:COLL:PORT1:SHORT3:L3 2.0E0
:SENS1:CORR:COLL:PORT1:SHORT3:L3?
```

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:LABEL <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:LABEL?

Description: Sets the Short3 standard label on the selected port of the indicated channel. Outputs the Short1 standard label on the selected port of the indicated channel.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example:
```
:SENS1:CORR:COLL:PORT1:SHORT3:LABEL 'IC7000'
:SENS1:CORR:COLL:PORT1:SHORT3:LABEL?
```

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:OFFS <NRf>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:OFFS?

Description: Sets the Short3 standard offset on the selected port of the indicated channel. Outputs the Short3 standard offset on the selected port of the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example:
```
:SENS1:CORR:COLL:PORT1:SHORT3:OFFS 2.5E1
:SENS1:CORR:COLL:PORT1:SHORT3:OFFS?
```

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:SERial <string>
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:SERial?

Description: Sets the Short3 standard serial number on the selected port of the indicated channel. Outputs the Short1 standard serial number on the selected port of the indicated channel.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:PORT1:SHORT3:SER '123456'**

:SENS1:CORR:COLL:PORT1:SHORT3:SER?

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD1

Description: Initiates collection of the Sliding Load1 standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SLOAD1**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD2

Description: Initiates collection of the Sliding Load2 standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SLOAD2**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD3

Description: Initiates collection of the Sliding Load3 standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SLOAD3**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD4

Description: Initiates collection of the Sliding Load4 standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SLOAD4**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD5

Description: Initiates collection of the Sliding Load5 standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SLOAD5**

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD6

Description: Initiates collection of the Sliding Load6 standard data for the calibration on the given port. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:SLOAD6**

5-48 :SENSe{1-16}:CORRection:COLLect Subsystem

The :SENSe{1-16}:CORRection:COLLect subsystem sets various coefficients and parameters for a pending calibration.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:COLLect:REFerence:Z0 <NRf>
:SENSe{1-16}:CORRection:COLLect:REFerence:Z0?

Description: Sets the Z0 (Z zero) reference impedance to use for the calibration. Outputs the reference impedance to use for the calibration.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: 1E-4 to 1E10

Default Value: 5.00000000000E+001

Syntax Example: **:SENS:CORR:COLL:REF:Z0 7.5E1**

:SENS:CORR:COLL:REF:Z0?

:SENSe{1-16}:CORRection:COLLect:SAVe

Description: Initiates calculation of correction coefficients for the programmed calibration type. The command should only be initiated if all required measurements have been collected. If the command is issued before all measurements are completed, an error is generated. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:SAV**

:SENSe{1-16}:CORRection:COLLect:TFR:CLEar

Description: Clears the list of Transmission Frequency Response calibrations. No query.

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:TFR:CLE**

5-49 :SENSe{1-16}:CORRection:COLLect[:CALa]:PORT

These commands set a 2-Port Calibration.

Calibration Subsystems with Actual Calibration

Related calibration subsystems that perform actual calibrations are:

- ":SENSe{1-16}:ABORtcal Subsystem" on page 5-136
- ":SENSe{1-16}:CORRection:COLLect:PORT Subsystem" on page 5-157
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189

Calibration Abbreviations

The calibration abbreviations and their calibration types are:

- :1P2PF refers to a one-path two-port calibration forward direction
- :1P2PR refers to a one path two port calibration reverse direction
- :FULL1 refers to a full one port calibration
- :FULL2 refers to a full two port calibration
- :FULLB refers to a full one port reflection calibration on both ports
- :RESP1 refers to a one port response calibration
- :RESPB refers to a one port response calibration both ports
- :TFRB refers to a transmission frequency response calibration both directions
- :TFRF refers to a transmission frequency response calibration forward direction
- :TFRR refers to a transmission frequency response calibration reverse direction

Each calibration simulation type command is described in greater detail in the individual command descriptions below.

Most calibration commands of this type do not have a directly related query. To query the state of these commands, use:

```
:SENSe{1-16}:CORRection:COLLect:TYPe?
```

The :SENSe{1-16}:CORRection:COLLect:PORT subsystem commands start an actual instrument calibration.

:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:1P2PF

Description: Sets the first calibration (:CALA) to One-Path Two-Port Forward calibration. No query.

If a second 2-Port Calibration (:CALB) is required, use:

```
:SENSe{1-16}:CORRection:COLLect:CALB:1P2PF
```

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **`:SENS:CORR:COLL:PORT1:1P2PF`**

:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:1P2PR

Description: Sets the first Two-Port calibration to One-Path Two-Port Reverse on the indicated port pair.

No query

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:1P2PR**

:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:FULL2

Description: Sets the first Two-Port calibration to Full Two-Port on the indicated port pair. No query

Cmd Parameters: NA

Range: NA

Default: NA

Syntax Example: **:SENS:CORR:COLL:PORT1:FULL2**

5-50 :SENSe{1-16}:CORRection:COLLect Subsystem

The :SENSe{1-16}:CORRection:COLLect subsystem commands start an actual calibration and limited to Port1 and/or Port 2.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

Related Command Subsystems

These commands are similar to those in the :SENSe{1-16}:CORRection:COLLect[:CALa]:PORT Subsystem above where any port pair on a 2-Port VNA can be specified.

- :SENSe{1-16}:CORRection:COLLect[:CALa]:PORT on page 5-189

Calibration Method Names

Related calibration types use the same command structure of:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:XXXXX
```

where the calibration XXXXX type is one of the following:

- 1P2PF = One-path two-port calibration, forward direction
- 1P2PR = One path two port calibration, reverse direction
- FULL1 = Full one port calibration
- FULL2 = Full two port calibration
- FULLB = Full one port reflection calibration, both ports
- RESP1 = One port response calibration
- RESPB = One port response calibration, both ports
- TFRB = Transmission frequency response calibration, both directions
- TFRF = Transmission frequency response calibration, forward direction
- TFRR = Transmission frequency response calibration, reverse direction

Each calibration type command is described in greater detail in the following sections. Related to the calibration commands are calibration simulation commands in the general form of :SENSe{1-16}:CORRection:COEFFicient[:METHod]:XXXXX that use the same abbreviations above.

To query the state of this calibration command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use the following command with the appropriate abbreviation above:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:XXXXX
```

:SENSe{1-16}:CORRection:COLLect[:METHod]:1P2PF

Description: Selects One-Path Two-Port Forward calibration (1P2PF) as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this calibration command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:1P2PF
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:1P2PF**

:SENSe{1-16}:CORRection:COLLect[:METHod]:1P2PR

Description: Selects One-Path Two-Port Reverse calibration (1P2PR) as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:1P2PR
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:1P2PR**

:SENSe{1-16}:CORRection:COLLect[:METHod]:FULL1

Description: Selects full one port reflection calibration as the calibration type. After the method is set, the calibration must be performed. No query.

Before sending this command, the simulation port must be specified using:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT
```

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:FULL1
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:FULL1**

:SENSe{1-16}:CORRection:COLLect:FULL2

Description: Selects full two port calibration as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:FULL2
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:FULL2**

:SENSe{1-16}:CORRection:COLLect:FULLB

Description: Select full one port reflection calibration both ports as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:FULLB
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:FULLB**

:SENSe{1-16}:CORRection:COLLect[:METHod]:LINE <char>
:SENSe{1-16}:CORRection:COLLect[:METHod]:LINE?

Description: Select the line type for calibration. Outputs the line type for calibration.

See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration comp onents, connectors, and their command parameters.

Cmd Parameters: <char> COAXial | MICROstrip | NONDISpersive | WAVEguide"

Query Parameters: <char> COAX | MICRO | NONDIS | WAVE

Range: NA

Default Value: COAX

Syntax Example: **:SENS:CORR:COLL:LINE COAX**

:SENS:CORR:COLL:LINE?

:SENSe{1-16}:CORRection:COLLect[:METHod]:LOAD <char>
:SENSe{1-16}:CORRection:COLLect[:METHod]:LOAD?

Description: Selects the load type broadband/sliding for calibration. Outputs the load type selection broadband/sliding for calibration.

Cmd Parameters: <char> FIXed | SLIDing

Query Parameters: <char> FIX | SLID

Range: NA

Default Value: FIX

Syntax Example: **:SENS:CORR:COLL:LOAD FIX**

:SENS:CORR:COLL:LOAD?

:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT <char>
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT?

Description: Sets the calibration port for a RESP1 or FULL1 calibration. Queries the calibration ports or port pairs. Outputs character data for the port combinations. Several examples of various calibration setups are described in the subsections below.

Reflection Frequency Response Calibration

A Reflection Frequency Response Calibration can have up to four reflection frequency response calibrations in one session.

Transmission Frequency Response Calibration

A Transmission Frequency Response Calibration can have up to six transmission frequency response calibrations picked from these types: TFRF, TFRR and TFRB.

One-Port Calibration

A One-Port Calibration can consist of up to four, FULL1 calibrations in one session.

Two-Port Calibration

A Two-Port Calibration can consist of one or two calibrations picked from the following types:

- FULL2, 1P2PF, and 1P2PR.
- The port pair assignments must be independent.
- Both Reflection and Thru measurements can provide FULL2, 1P2PF, and 1P2PR.

Cmd Parameters: <char> PORT1 | PORT2 | PORT12

Query Parameters: <char> PORT1 | PORT2 | PORT12

Range: NA

Default Value: PORT12

Syntax Example: **:SENS:CORR:COLL:PORT PORT2**

:SENS:CORR:COLL:PORT?

:SENSe{1-16}:CORRection:COLLect[:METHod]:RESP1

Description: Selects One-Port Response Calibration as the calibration type. After the method is set, the calibration must be performed. No query.

Before sending this command, the simulation port must be specified using:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT
```

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:RESP1
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:SENS:CORR:COLL:RESP1`**

:SENSe{1-16}:CORRection:COLLect[:METHod]:REFTHru

Description: Selects the Reflection and Scalar Thru calibration as the calibration type (MS46121A/B only). After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:RESPB
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:SENS:CORR:COLL:RESPB`**

:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRB

Description: Selects transmission frequency response calibration both directions as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:TFRB
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:SENS:CORR:COLL:TFRB`**

:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRF

Description: Selects transmission frequency response calibration forward direction as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:TFRF
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:TFRF**

:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRR

Description: Selects transmission frequency response calibration reverse direction as the calibration type. After the method is set, the calibration must be performed. No query.

To query the state of this command, use:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?
```

To simulate this calibration, use:

```
:SENSe{1-16}:CORRection:COEFFicient[:METHod]:TFRR
```

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:CORR:COLL:TFRR**

:SENSe{1-16}:CORRection:COLLect:THRu:ADD <char>

Description: Adds Transmission Throughs (Thrus) to the selected 3-Port Calibration or 4-Port Calibration on the indicated channel. Adds Scalar Throughs (Thrus) to the selected Reflection and Scalar Thru Calibration on the indicated channel. No query. The available throughs (thrus) are:

- THRu12 = Through line between Port 1 and Port 2; short form is THR12..

Cmd Parameters: <char> THRu12

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:THR:ADD THRU12**

:SENSe{1-16}:CORRection:COLLect:THRu:CLEar

Description: Clears the Transmission Thrus of the selected Three-Port Calibration or Four-Port Calibration on the indicated channel. No Query

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:THR:CLE**

:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe?

Description: Query only. Outputs the calibration types. Several examples of various calibration setups are described below.

Reflection Frequency Response Calibration

A Reflection Frequency Response Calibration can have up to four reflection frequency response calibrations in one session. The following is a return for the calibration type query for a response calibration on port 2:

```
RESP1,RESP1,RESP1
```

Transmission Frequency Response Calibration

A Transmission Frequency Response Calibration can have up to six transmission frequency response calibrations picked from these types: TFRF, TFRR and TFRB. The following is the return for the calibration type query for a mix of transmission response calibrations on all possible port pairs:

```
TFRF,TFRR,TFRB,TFRR,TFRF,TFRB
```

One-Port Calibration

A One-Port Calibration can consist of up to four, FULL1 calibrations in one session. The following is a return for the calibration type query for a FULL1 calibration:

```
FULL1,FULL1,FULL1
```

Two-Port Calibration

A Two-Port Calibration can consist of one or two calibrations picked from the following types: FULL2, 1P2PF, and 1P2PR. The port pair assignments must be independent. The Calibration Methods can be Reflection and Thru measurements. Both Reflection and Thru measurements can provide FULL2, 1P2PF, and 1P2PR. The following is a typical return for the calibration type query:

```
1P2PF,FULL2
```

Response Calibration Both and Full Calibration Both

The RESPB and FULLB calibration types return the command arguments shown below:

```
RESPB returns RESP1,RESP1
FULLB returns FULL1,FULL1
```

Set of Query Calibration Ports

Use the following commands to set or query the calibration ports:

```
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT?
```

Query Parameters: <char>{, <char2>}{, <char3>, <char4>, <char5>, <char6>}

Where the <char> values are combinations of:

- RESP1
- FULL1
- FULL2
- TFRF
- TFRR
- TFRB
- 1P2PF

- 1P2PR

Range: NA

Default Value: FULL2

Syntax Example: **:SENS:CORR:COLL:TYP?**

5-51 :SENSe{1-16}:CORRection:COLLect:WAVeguide

The :SENSe{1-16}:CORRection:COLLect:WAVeguide subsystem sets the calibration parameters and coefficients for waveguide line types. Use this subsystem to set waveguide parameters before starting a calibration.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MICrostrip Subsystem" on page 5-152
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:COLLect:WAVeguide:DIELectric <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:DIELectric?

Description: Sets the waveguide calibration kit dielectric value on the indicated channel. Outputs the waveguide calibration kit dielectric value on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR3> The output parameter is a unitless number.

Range: MPND

Default Value: 1.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:DIEL 1.2E0**

:SENS1:CORR:COLL:WAV:DIEL?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:FREQuency <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:FREQuency?

Description: Sets the waveguide calibration kit cutoff frequency on the indicated channel. Outputs the waveguide calibration kit cutoff frequency on the indicated channel.

Cmd Parameters: <NRf> The input parameter is Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: **:SENS1:CORR:COLL:WAV:FREQ 5.0E9**

:SENS1:CORR:COLL:WAV:FREQ?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:KIT <char>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:KIT?

Description: Sets the waveguide kit type on the indicated channel. Outputs the waveguide kit type on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <char> WR10 | WR12 | WR15 | USER1 | USER2 | USER3 | USER4 | USER5 | USER6 | USER7 | USER8

Where:

- WR28 = Typical WR-28 RF waveguide
- WR42 = Typical WR-42 RF waveguide
- WR64 = Typical WR-164 RF waveguide
- USERx = User defined waveguide x. The device can be renamed by the user but programmatically is always referred to as "USERx":

Where x is any number between 1 and 8.

- USER1 = User defined waveguide 1. The device can be renamed by the user but programmatically is always referred to as "USERx":

Query Parameters: <char> WR28 | WR42 | WR64 | WR75 | WR90 | WR112 | WR137 | WR159 | WR187 | WR229 | USER1 | USER2 | USER3 | USER4 | USER5 | USER6 | USER7 | USER8

Range: NA

Default Value: USER 1

Syntax Example: **:SENS1:CORR:COLL:WAV:KIT WR28**

:SENS1:CORR:COLL:WAV:KIT?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:LABel <string>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LABel?

Description: Sets the waveguide calibration kit label on the indicated channel. Outputs the waveguide calibration kit label on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters.

See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: WR10

Syntax Example: **:SENS1:CORR:COLL:WAV:LAB 'wave1'**

:SENS1:CORR:COLL:WAV:LAB?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:L0 <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:L0?

Description: Sets the waveguide calibration kit L0 (L zero) load inductance on the indicated channel. Outputs the waveguide calibration kit load inductance on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Henrys.

Cmd Parameters: <NR3> The output parameter is in Henrys.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:LOAD:L0 7.5E1**

:SENS1:CORR:COLL:WAV:LOAD:L0?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:OFFSet <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:OFFSet?

Description: Sets the waveguide calibration kit load offset on the indicated channel. Outputs the waveguide calibration kit load offset on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:LOAD:OFFS 2.5E1**

:SENS1:CORR:COLL:WAV:LOAD:OFFS?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:R <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:R?

Description: Sets the waveguide calibration kit load resistance on the indicated channel. Outputs the waveguide calibration kit load resistance on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Ohms.

Query Parameters: <NR3> The output parameter is in Ohms.

Range: MPND

Default Value: 5.00000000000E+001

Syntax Example: **:SENS1:CORR:COLL:WAV:LOAD:R 7.5E1**

:SENS1:CORR:COLL:WAV:LOAD:R?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C0 <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C0?

Description: Sets the waveguide calibration kit open capacitance C0 (C zero) value on the indicated channel. The C0 coefficient is measured in Farads. Outputs the waveguide calibration kit open capacitance C0 value on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Farads.

Query Parameters: <NR3> The output parameter is in Farads.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:OPEN:C0 3.0E-12**

:SENS1:CORR:COLL:WAV:OPEN:C0?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C1 <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C1?

Description: Sets the waveguide calibration kit open capacitance C1 (C one) term on the indicated channel. The C1 coefficient is measured in Farads/Hertz. Outputs the waveguide calibration kit open capacitance1 term on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz.

Query Parameters: <NR3> The output parameter is in Farads/Hertz.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:OPEN:C1 3.0E-12**

:SENS1:CORR:COLL:WAV:OPEN:C1?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C2 <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C2?

Description: Sets the waveguide calibration kit open capacitance C2 term on the indicated channel. The C2 coefficient is measured in Farads/Hertz^2. Outputs the waveguide calibration kit open capacitance C2 term on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^2.

Query Parameters: <NR3> The output parameter is in Farads/Hertz^2.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:OPEN:C2 3.0E-12**

:SENS1:CORR:COLL:WAV:OPEN:C2?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C3 <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C3?

Description: Sets the waveguide calibration kit open capacitance C3 term on the indicated channel. The C3 coefficient is measured in Farads/Hertz^3. Outputs the waveguide calibration kit open capacitance C3 term on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Farads/Hertz^3.

Query Parameters: <NR3> The output parameter is in Farads/Hertz^3.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:OPEN:C3 3.0E-12**

:SENS1:CORR:COLL:WAV:OPEN:C3?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:OFFSet <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:OFFSet?

Description: Sets the waveguide calibration kit open offset on the indicated channel. Outputs the waveguide calibration kit open offset on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example: **:SENS1:CORR:COLL:WAV:OPEN:OFFS 1.2E-4**

:SENS1:CORR:COLL:WAV:OPEN:OFFS?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:SERial <string>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SERial?

Description: Sets the waveguide calibration kit serial number on the indicated channel. Outputs the waveguide calibration kit serial number on the indicated channel.

Cmd Parameters: <string> The input parameter is any combination of numbers and letters. See definition of "<string>" on page 2-10.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: NA

Syntax Example: **:SENS1:CORR:COLL:WAV:SER '12012008'**

:SENS1:CORR:COLL:WAV:SER?

:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT1:OFFSet <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT1:OFFSet?

Description: Sets the waveguide calibration kit short1 offset on the indicated channel. Outputs the waveguide calibration kit short1 offset on the indicated channel.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:WAV:SHORT1:OFFS 1.2E-4`

`:SENS1:CORR:COLL:WAV:SHORT1:OFFS?`

:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT2:OFFSet <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT2:OFFSet?

Description: Sets the waveguide calibration kit short2 offset on the indicated channel. Outputs the waveguide calibration kit label short2 offset on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:WAV:SHORT2:OFFS 5.0E-9`

`:SENS1:CORR:COLL:WAV:SHORT2:OFFS?`

:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT3:OFFSet <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT3:OFFSet?

Description: Sets the waveguide calibration kit short3 offset length on the indicated channel. Outputs the waveguide calibration kit short3 offset length on the indicated channel. See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Meters.

Query Parameters: <NR3> The output parameter is in Meters.

Range: MPND

Default Value: See "Calibration Component Parameters" on page 2-31.

Syntax Example: `:SENS1:CORR:COLL:WAV:SHORT3:OFFS 5.0E9`

`:SENS1:CORR:COLL:WAV:SHORT3:OFFS?`

:SENSe{1-16}:CORRection:COLLect:WAVeguide:SLOAD:MINF <NRf>
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SLOAD:MINF?

Description: Sets the waveguide calibration kit sliding load minimum frequency on the indicated channel. Outputs the waveguide calibration kit sliding load minimum frequency on the indicated channel.See "Calibration Component Parameters" on page 2-31 for a complete listing of calibration components, connectors, and their Command Parameters.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: MPND

Default Value: 2.00000000000E+009

Syntax Example: **:SENS1:CORR:COLL:WAV:SLOAD:MINF 10.0E7**

```
:SENS1:CORR:COLL:WAV:SLOAD:MINF?
```

5-52 :SENSe{1-16}:CORRection:EXTension Subsystem

The :SENSe{1-16}:CORRection:EXTension subsystem commands control the reference plane extension from the test ports.

Time Domain, Group Delay, and Reference Plane Subsystems

Related time domain, group delay, and reference place subsystems are:

- Section 5-21 :CALCulate{1-16}:REFerence Subsystem on page 5-59
- Section 5-32 :CALCulate{1-16}[:SELected]:TRANsform:TIME Subsystem on page 5-98
- Section 5-52 :SENSe{1-16}:CORRection:EXTension Subsystem on page 5-206

:SENSe{1-16}:CORRection:EXTension:PORT{1-2} <NRf>
:SENSe{1-16}:CORRection:EXTension:PORT{1-2}?

Description: Sets the reference plane extension for the indicated port. Outputs the reference plane extension for the indicated port.Also computes the distance value and shows it in the Distance field in the Reference Plane menu in the user interface

Cmd Parameters: <NRf> The input parameter is in seconds.

Query Parameters: <NR3> The output parameter is in seconds.

Range: MPND

Default Value: 0.00000000000E+000

Syntax Example:
```
:SENS:CORR:EXT:PORT1 3.0E-3
:SENS:CORR:EXT:PORT1?
```

5-53 :SENSe{1-16}:CORRection:ISOLation Subsystem

The :SENSe{1-16}:CORRection:ISOLation subsystem command controls the use of the isolation data during calibration.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:ISOLation:STATe <char>
:SENSe{1-16}:CORRection:ISOLation:STATe?

Description: Toggles the use of the isolation coefficient data on/off during correction. Outputs the on/off status of the use of the isolation coefficients during correction.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:
```
:SENS:CORR:ISOL:STAT ON
:SENS:CORR:ISOL:STAT?
```

5-54 :SENSe{1-16}:CORRection:STATe Subsystem

The :SENSe{1-16}:CORRection:STATe subsystem commands controls the RF correction.

Calibration Setup Subsystems

These subsystems are used during various phases of calibration configuration setup:

- ":CALCulate{1-16}:REFerence Subsystem" on page 5-59
- ":SENSe{1-16}:CORRection:COLLect:METHod Subsystem" on page 5-151
- ":SENSe{1-16}:CORRection:COLLect:MULTIple Subsystem" on page 5-156
- ":SENSe{1-16}:CORRection:COLLect Subsystem" on page 5-188
- ":SENSe{1-16}:CORRection:COLLect[:CALa]:PORT" on page 5-189
- ":SENSe{1-16}:CORRection:ISOLation Subsystem" on page 5-207
- ":SENSe{1-16}:CORRection:STATe Subsystem" on page 5-208

:SENSe{1-16}:CORRection:STATe <char>
:SENSe{1-16}:CORRection:STATe?

Description: Toggles RF correction on/off. Outputs the RF correction on/off status.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:
```
:SENS:CORR:STAT ON
:SENS:CORR:STAT?
```

5-55 :SENSe{1-16}:FREQuency Subsystem

The :SENSe{1-16}:FREQuency subsystem commands control the various instrument frequencies.

Sweep Subsystems

Related sweep configuration and control subsystems are:

- Section 5-38 :SENSe{1-16}:AVERage Subsystem on page 5-136
- Section 5-55 :SENSe{1-16}:FREQuency Subsystem on page 5-209
- Section 5-63 :SENSe{1-16}:SWEep Subsystem on page 5-236

Frequency Limits

The frequency limits for the :SENSe{1-16}:FREQuency subsystem and other commands are described in detail in Chapter 1 "General Information", "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 and the related tables in that section. In general, the frequency default values and limits are affected by the instrument model number.

:SENSe{1-16}:FREQuency:CENTer <NRf>
:SENSe{1-16}:FREQuency:CENTer?

Description: Sets the center value of the sweep range. Outputs the center value of the sweep range.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Default Value: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Syntax Example:
```
:SENS:FREQ:CENT 2.0E9
:SENS:FREQ:CENT?
```

:SENSe{1-16}:FREQuency:CW <NRf>
:SENSe{1-16}:FREQuency:CW?

Description: Sets the CW frequency. Outputs the CW frequency.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13

Default Value: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Syntax Example:
```
:SENS:FREQ:CW 1.0E9
:SENS:FREQ:CW?
```

:SENSe{1-16}:FREQuency:DATA <block>
:SENSe{1-16}:FREQuency:DATA?

Description: Enters a new frequency list. Outputs the frequency list.

Cmd Parameters: See definition of "<block> or <arbitrary block>" on page 2-10.

Query Parameters: See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:SENS:FREQ:DATA <block>**

:SENS:FREQ:DATA?

:SENSe{1-16}:FREQuency:SPAN <NRf>
:SENSe{1-16}:FREQuency:SPAN?

Description: Sets the span value of the sweep range. Outputs the span value of the sweep range.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13

Default Value: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Syntax Example: **:SENS:FREQ:SPAN 5.0E9**

:SENS:FREQ:SPAN?

:SENSe{1-16}:FREQuency:STARt <NRf>
:SENSe{1-16}:FREQuency:STARt?

Description: Sets the start value of the sweep range. Outputs the start value of the sweep range.

Cmd Parameters: <NRf> The input parameter is in Hertz, Meters, or Seconds.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13

Default Value: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Syntax Example: **:SENS:FREQ:STAR 2.0E9**

:SENS:FREQ:STAR?

:SENSe{1-16}:FREQuency:STOP <NRf>
:SENSe{1-16}:FREQuency:STOP?

Description: Sets the stop value of the sweep range. Outputs the stop value of the sweep range.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz, Meters, or Seconds.

Range: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Default Value: See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13.

Syntax Example: **:SENS:FREQ:STOP 8.0E9**

:SENS:FREQ:STOP?

5-56 :SENSe{1-16}:FSEGMent Subsystem

The :SENSe{1-16}:FSEGMent subsystem commands are used to configure the active frequency-based segment.

Limit Line and Segment Subsystems

Related limit line and segment configuration and control subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107
- ":SENSe{1-16}:FSEGMent Subsystem" on page 5-211.
- ":SENSe{1-16}:FSEGMent{1-50} Subsystem" on page 5-218.
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231
- ":SENSe{1-16}:SEGMent Subsystem" on page 5-235

:SENSe{1-16}:FSEGMent:ADD

Description: Adds a new frequency-sweep segment at the end of the frequency-based segment table. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz. No query.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: NA

Range: The range depends on the segment sequence and location:

- Minimum Segment Range = 2 Hz
- Minimum Segment Points = 2 points
- For first segment, Minimum Segment Frequency = Minimum Instrument Frequency.
- For highest frequency entered, Maximum Segment Frequency = Maximum Instrument Frequency.
- For the first segment entered, the Maximum Segment Range = Maximum Instrument Range = (Maximum Instrument Frequency minus Minimum Instrument Frequency).

Default Value: NA

Syntax Example: **:SENS:FSEGM:ADD**

:SENSe{1-16}:FSEGMent:CLEar

Description: Clears all currently defined segments from the frequency-based segment table, leaving a default segment. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:FSEGM:CLE**

:SENSe{1-16}:FSEGMent:COUNt?

Description: Query only. Outputs the number of segments in the frequency-based segmented sweep.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 50

Default Value: 1

Syntax Example: **:SENS:FSEGM:COUN?**

:SENSe{1-16}:FSEGMent:CWMODe[:STATe] <char>
:SENSe{1-16}:FSEGMent:CWMODe[:STATe]?

Description: Sets the CW mode on/off state in the last frequency-based segment being defined. If CW mode is set, segment has only 1 point. Outputs the CW mode on/off state in the last frequency-based segment being defined.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:SENS:FSEGM:CWMOD ON**

:SENS:FSEGM:CWMOD?

:SENSe{1-16}:FSEGMent:DATA?

Description: Query only. Outputs the frequency-based segmented sweep table.

Query Parameters: See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:SENS:FSEGM:DATA?**

:SENSe{1-16}:FSEGMent:DISPlay <char>
:SENSe{1-16}:FSEGMent:DISPlay?

Description: Sets the frequency/index display mode for the frequency-based segmented sweep. Outputs the frequency/index display mode for the frequency-based segmented sweep.

Cmd Parameters: <char> FREQbase | INDEXbase

Query Parameters: <char> FREQ | INDEX

Range: NA

Default Value: FREQ

Syntax Example: **:SENS:FSEGM:DISP FREQ**

:SENS:FSEGM:DISP?

:SENSe{1-16}:FSEGMent:FREQuency:ACTive:STARt?

Description: Query only. Outputs the start frequency of the first active frequency-based segment. The output result is affected the by segment on/off status. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to (Maximum Instrument Frequency minus Minimum Frequency Step Size)

Default Value: The default value depends on the installed options.

Syntax Example: **:SENS:FSEGM:FREQ:ACT:STAR?**

:SENSe{1-16}:FSEGMent:FREQuency:ACTive:STOP?

Description: Query only. Outputs the stop frequency of the last active frequency-based segment. The output result is affected the by segment on/off status. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: (Minimum Instrument Frequency + Minimum Frequency Step Size) to Maximum Instrument Frequency

Default Value: Maximum Instrument Frequency

Syntax Example: **:SENS:FSEGM:FREQ:ACT:STOP?**

:SENSe{1-16}:FSEGMent:FREQuency:FSTEp <NRf>
:SENSe{1-16}:FSEGMent:FREQuency:FSTEp?

Description: Sets the Segment Frequency Step Size in the last frequency-based segment being defined. Outputs the Segment Frequency Step Size in the last frequency-based segment being defined. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Frequency Step Size to (Maximum Instrument Frequency minus Minimum Instrument Frequency)

Default Value: The default value depends on installed options.

Syntax Example: **:SENS:FSEGM:FREQ:FSTE 10.0E3**

:SENS:FSEGM:FREQ:FSTE?

:SENSe{1-16}:FSEGMent:FREQuency:FSTOp <NRf>
:SENSe{1-16}:FSEGMent:FREQuency:FSTOp?

Description: Sets the Segment Stop Frequency in the last frequency-based segment being defined. Outputs the Segment Stop Frequency in the frequency-based segment being defined. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: (Minimum Instrument Frequency + Minimum Frequency Step Size) to Maximum Instrument Frequency

Default Value: Maximum Instrument Frequency

Syntax Example: **:SENS:FSEGM:FREQ:FSTO 10.0E9**

:SENS:FSEGM:FREQ:FSTO?

:SENSe{1-16}:FSEGMent:FREQuency:STARt <NRf>
:SENSe{1-16}:FSEGMent:FREQuency:STARt?

Description: Sets the Segment Start Frequency of the frequency-based segmented sweep. Outputs the Segment Start Frequency of the frequency-based segmented sweep. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to (Maximum Instrument Frequency minus Minimum Frequency Step Size)

Default Value: Default Start Frequency (for normal sweeps, i.e. 300kHz here)

Syntax Example: **:SENS:FSEGM:FREQ:STAR 10.0E9**

:SENS:FSEGM:FREQ:STAR?

:SENSe{1-16}:FSEGMent:FREQuency:STOP <NRf>
:SENSe{1-16}:FSEGMent:FREQuency:STOP?

Description: Sets the Segment Stop Frequency of the frequency-based segmented sweep. Outputs the Segment Stop Frequency of the frequency-based segmented sweep. Available range is limited by the range of the existing segments. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: (Minimum Instrument Frequency + Minimum Frequency Step Size) to Maximum Instrument Frequency

Default Value: Maximum Instrument Frequency

Syntax Example: **:SENS:FSEGM:FREQ:STOP 20.0E9**

:SENS:FSEGM:FREQ:STOP?

:SENSe{1-16}:FSEGMent:FREQuency[:CW][:FIXed] <NRf>
:SENSe{1-16}:FSEGMent:FREQuency[:CW][:FIXed]?

Description: Sets the CW Segment Frequency in the last frequency-based segment being defined. Outputs the CW Segment Frequency in the frequency-based segment being defined. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR1> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency

Default Value: 70000

Syntax Example: **:SENS:FSEGM:FREQ 10.0E6**

:SENS:FSEGM:FREQ?

:SENSe{1-16}:FSEGMent:MAXPoints?

Description: Query only. Outputs the total number of sweep points in the frequency-based segments. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Query Parameters: <NR1> The output parameter is an integer.

Range: Range depends on if CW mode is set:

- If CW is set, range equals 1 (one) point.
- If in sweep mode or non-CW mode, range is from 2 (two) points to Maximum Instrument Points.

Default Value: 15

Syntax Example: **:SENS:FSEGM:MAXP?**

:SENSe{1-16}:FSEGMent:SPAntype <char>
:SENSe{1-16}:FSEGMent:SPAntype?

Description: Sets the Segment Span Type of the last frequency-based segment being defined to the specified span type. Outputs the Segment Span Type of the last frequency-based segment being defined.

STARTSTOP Selected

If STARTSTOP is selected, each segment is defined by the:

- Segment Start Frequency
- Segment Stop Frequency
- Number of Segment Points

STARTSTEP Selected

If STARTSTEP is selected, each segment is defined by the:

- Start Segment Frequency
- Frequency Step Size
- Number of Segment Points

Cmd Parameters: <char> STARTSTOP | STARTSTEP

Query Parameters: <char> STARTSTOP | STARTSTEP

Range: NA

Default Value: STARTSTOP

Syntax Example: **:SENS:FSEGM:SPA STARTSTOP**

:SENS:FSEGM:SPA?

:SENSe{1-16}:FSEGMent:SWEep:MAXimize

Description: Maximizes the frequency range of the frequency-based segmented sweep. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:FSEGM:SWE:MAX**

:SENSe{1-16}:FSEGMent:SWEep:POINt <NR1>
:SENSe{1-16}:FSEGMent:SWEep:POINt?

Description: Sets the number of Segment Sweep Points in the last frequency-based segment being defined. Outputs the number of sweep points in the last frequency-based segment being defined. If frequency-sweep is set, the range is from 2 (two) points to the Maximum Instrument Points. If the separation between the segment start and stop frequencies is 1 Hz, the number of points is 2 (two). If CW is set, the number of points is 1 (one). For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NR1> The input parameter is an integer.

Query Parameters: <NR1> The output parameter is an integer.

Range: The range depends on if the CW mode is set:

- If CW is set, range = 1 (one) point.
- If in sweep mode or non-CW mode) range = from 2 (two) points to Maximum Instrument Points.

Default Value: 15 or 1 depending on span type.

Syntax Example: **:SENS:FSEGM:SWE:POIN 1.01E2**

:SENS:FSEGM:SWE:POIN?

:SENSe{1-16}:FSEGMent[:STATe] <char>
:SENSe{1-16}:FSEGMent[:STATe]?

Description: Toggles the on/off state of the last frequency-based segment being defined. Outputs the on/off state of the last frequency-based segment being defined.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 1

Syntax Example: **:SENS:FSEGM ON**

:SENS:FSEGM?

5-57 :SENSe{1-16}:FSEGMent{1-50} Subsystem

The :SENSe{1-16}:FSEGMent{1-50} subsystem commands are used to configure the indicated frequency-based segment.

Limit Line and Segment Subsystems

Related limit line and segment configuration and control subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107
- ":SENSe{1-16}:FSEGMent Subsystem" on page 5-211.
- ":SENSe{1-16}:FSEGMent{1-50} Subsystem" on page 5-218.
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231
- ":SENSe{1-16}:SEGMent Subsystem" on page 5-235

:SENSe{1-16}:FSEGMent{1-50}:CWMODe[:STATe] <char>
:SENSe{1-16}:FSEGMent{1-50}:CWMODe[:STATe]?

Description: Sets the CW mode on/off state in the indicated frequency-based segment. Outputs the CW mode on/off state in the indicated frequency-based segment.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:SENS:FSEGM1:CWMOD ON**

:SENS:FSEGM1:CWMOD?

:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTArt <NRf>
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTArt?

Description: Sets the Segment Start Frequency in the indicated frequency-based segment. Outputs the Segment Start Frequency in the indicated frequency-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Query Parameters: <NRf> The input parameter is in Hertz.

Cmd Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to (Maximum Instrument Frequency minus Minimum Frequency Step Size)

Default Value: Default Start Frequency (for normal sweeps, i.e. 300kHz here)

Syntax Example: **:SENS:FSEGM1:FREQ:FSTA 3.0E9**

:SENS:FSEGM1:FREQ:FSTA?

:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTEp <NRf>
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTEp?

Description: Sets the Segment Step Size in the indicated frequency-based segment. Outputs the frequency step size (Fstep) in the indicated frequency-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Frequency Step Size to (Maximum Instrument Frequency minus Minimum Instrument Frequency)

Default Value: The default value depends on the installed options.

Syntax Example: **:SENS:FSEGM1:FREQ:FSTE 1.00E4**

:SENS:FSEGM1:FREQ:FSTE?

:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTOp <NRf>
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTOp?

Description: Sets the Segment Stop Frequency in the indicated frequency-based segment. Outputs the Segment Stop Frequency in the indicated frequency-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: (Minimum Instrument Frequency + Minimum Frequency Step) to Maximum Instrument Frequency

Default Value: Maximum Instrument Frequency

Syntax Example: **:SENS:FSEGM1:FREQ:FSTO 10.0E9**

:SENS:FSEGM1:FREQ:FSTO?

:SENSe{1-16}:FSEGMent{1-50}:FREQuency[:CW][:FIXed] <NRf>
:SENSe{1-16}:FSEGMent{1-50}:FREQuency[:CW][:FIXed]?

Description: Sets the Segment CW Frequency in the indicated frequency-based segment. Outputs the Segment CW Frequency in the indicated frequency-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency

Default Value: Default Start Frequency (for normal sweeps, i.e. 300kHz here)

Syntax Example: **:SENS:FSEGM1:FREQ 5.0E9**

:SENS:FSEGM1:FREQ?

:SENSe{1-16}:FSEGMent{1-50}:SPAntype <char>
:SENSe{1-16}:FSEGMent{1-50}:SPAntype?

Description: Sets the Segment Span Type of the indicated frequency-based segment. Outputs the Segment Span Type of the indicated frequency-based segment.

If STARTSTOP is selected, each segment is defined by the:

- Segment Start Frequency
- Segment Stop Frequency
- Number of Segment Points

If STARTSTEP is selected, each segment is defined by the:

- Start Segment Frequency
- Frequency Step Size
- Number of Segment Points

Cmd Parameters: <char> STARTSTOP | STARTSTEP

Query Parameters: <char> STARTSTOP | STARTSTEP

Range: NA

Default Value: STARTSTOP

Syntax Example: **:SENS:FSEGM1:SPA STARTSTOP**

:SENS:FSEGM1:SPA?

:SENSe{1-16}:FSEGMent{1-50}:SWEep:POINt <NR1>
:SENSe{1-16}:FSEGMent{1-50}:SWEep:POINt?

Description: Sets the Number of Segment Sweep Points in the indicated frequency-based segment. Outputs the number of sweep points in the indicated frequency-based segment.For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NR1> The input parameter is an integer.

Query Parameters: <NR1> The output parameter is an integer.

Range: Range depends on if CW mode is set.

- If CW is set, 1 point.
- If frequency-sweep is set, range is from 2 (two) points to Maximum Instrument Points.

Default Value: 15 or 1, depending on CW mode.

Syntax Example: **:SENS:FSEGM1:SWE:POIN 5.01E2**

:SENS:FSEGM1:SWE:POIN?

:SENSe{1-16}:FSEGMent{1-50}[:STATe] <char>
:SENSe{1-16}:FSEGMent{1-50}[:STATe]?

Description: Turns the indicated frequency-based segment on/off. Outputs the indicated frequency-based segment on/off state.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example:

```
:SENS:FSEGM1
:SENS:FSEGM1?
```

5-58 :SENSe{1-16}:HOLD Subsystem

The :SENSe{1-16}:HOLD subsystem command sets the hold function on a per-instrument basis.

If a channel number is not included in this command, the command will be applied to ALL channels.

Trigger, Hold, and External Source Subsystems

Related trigger, hold, and external source subsystems are:

- ":SENSe{1-16}:ABORtcal Subsystem" on page 5-136
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225

:SENSe{1-16}:HOLD:FUNCtion <char>
:SENSe{1-16}:HOLD:FUNCtion?

Description: Sets the hold function and provides these available options:

- CONTinuous = Perform continuous sweeps
- HOLD = Hold the sweep
- SINGle = Perform a single sweep and then hold

Outputs the hold status.

The operation of this command depends on the settings of the :SENSe{1-16}:HOLD:FUNCtion, :TRIGger[:SEQuence][:REMote]:SINGle, and :TRIGger[:SEQuence][:IMMediate][:REMote] commands. Each setting combination is described in the sections below.

:SENSe{1-16}:HOLD:FUNC CONT and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion CONTinuous

// Sweep State = The sweep is sweeping continuously

// Command Execution = The parser is ready for a command right
away.

:TRIGger[:SEQuence][:IMMediate][:REMote]

// Sweep State = The sweep restarts and sweeps continuously. When
the sweep gets to the end of the sweep, it continues to sweep.
There is NO STATUS information that the end of the sweep has been
reached.

// Command Execution = The parser is ready for a command right away
```

:SENSe{1-16}:HOLD:FUNC CONT and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion CONTinuous

// Sweep State = The sweep is sweeping continuously.

// Command Execution = The parser is ready for a command right away

:TRIGger[:SEQuence][:REMote]:SINGle

// Sweep State = The sweep restarts and sweeps continuously. When
the sweep gets to the end of the sweep, it sets the end of sweep
status bit and continues to sweep.

// Command Execution = Further execution is blocked until the end
of the sweep.

// Command Execution resumes when the sweep has reached the end of
the sweep.
```

:SENSe{1-16}:HOLD:FUNC HOLD and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion HOLD
// Sweep State = The sweep is stopped.
// Command Execution = The parser is ready for a command right away
:TRIGger[:SEQuence][:IMMediate][:REMote]
// Sweep State = The command has no effect. The sweep is stopped.
// Command Execution = The parser is ready for a command right away
```

:SENSe{1-16}:HOLD:FUNC HOLD and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion HOLD
// Sweep State = The sweep is stopped
// Command Execution = The parser is ready for a command right
away.
:TRIGger[:SEQuence][:REMote]:SINGle
// Sweep State = The sweep restarts and sweeps until the end of the
sweep, at which point it sets the end of sweep status bit and
stops.
// Command Execution = Further execution is blocked until the end
of the sweep.
// Command Execution resumes when the sweep has reached the end of
the sweep.
```

:SENSe{1-16}:HOLD:FUNC SING and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion SINGle
// Sweep State = The sweep does one complete sweep, goes into hold
and stops.
// Command Execution = The parser is ready for a command.
:TRIGger[:SEQuence][:IMMediate][:REMote]
// Sweep State = The command has no effect. The sweep is stopped.
// Command Execution = The parser is ready for a command.
```

:SENSe{1-16}:HOLD:FUNC SING and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion SINGle
// Sweep State = The sweep does one complete sweep, goes into hold
and stops.
// Command Execution = The parser is ready for a command right
away.
:TRIGger[:SEQuence][:REMote]:SINGle
// Sweep State = The sweep restarts and sweeps until the end of the
sweep, at which point it sets the end of sweep status bit and
stops.
// Command Execution = Further execution is blocked until the end
of the sweep.
```

```
// Command Execution resumes when the sweep has reached the end of
the sweep.
```

Cmd Parameters: <char> CONTinuous | HOLD | SINGle

Query Parameters: <char> CONT | HOLD | SING

Range: NA

Default Value: NA

Syntax Example: **:SENS:HOLD:FUNC CONT**

:SENS:HOLD:FUNC?

5-59 :SENSe{1-16}:ISEGMent Subsystem

The :SENSe{1-16}:ISEGMent subsystem commands are used to configure the active index-based segment.

To configure the index-based segments by segment number, use:

- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.

Limit Line and Segment Subsystems

Related limit line and segment configuration and control subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107
- ":SENSe{1-16}:FSEGMent Subsystem" on page 5-211.
- ":SENSe{1-16}:FSEGMent{1-50} Subsystem" on page 5-218.
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231
- ":SENSe{1-16}:SEGMent Subsystem" on page 5-235

:SENSe{1-16}:ISEGMent:ADD

Description: Adds a new segment at the end of the index-based segment table. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:ISEGM:ADD**

:SENSe{1-16}:ISEGMent:CLEar

Description: Clears all currently defined segments from the index-based segment table and adds a blank segment. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:ISEGM:CLE**

:SENSe{1-16}:ISEGMent:COUNt?

Description: Query only.

Outputs the number of segments in the index-based segmented sweep.

Query Parameters: <NR1> The output parameter is an integer.

Range: NA

Default Value: 1

Syntax Example: **:SENS:ISEGM:COUN?**

:SENSe{1-16}:ISEGMent:CWMODe[:STATe] <char>
:SENSe{1-16}:ISEGMent:CWMODe[:STATe]?

Description: Sets the CW mode on/off state in the index-based segment being defined. Returns the CW mode on/off state in the index-based segment being defined.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:SENS:ISEGM:CWMOD ON**

:SENS:ISEGM:CWMOD?

:SENSe{1-16}:ISEGMent:DATA?

Description: Query only.

Outputs the index-based segmented sweep table.

Query Parameters: See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:SENS:ISEGM:DATA?**

:SENSe{1-16}:ISEGMent:FREQuency:FSTArt <NRf>
:SENSe{1-16}:ISEGMent:FREQuency:FSTArt?

Description: Sets the start frequency in the index-based segment being defined. Outputs the start frequency in the index-based segment being defined. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency. Step Size = 0 Hz to Maximum Instrument Frequency.

Default Value: Default Start Frequency (for normal sweeps, i.e. 300kHz here)

Syntax Example: **:SENS:ISEGM:FREQ:FSTA 3.0E9**

:SENS:ISEGM:FREQ:FSTA?

:SENSe{1-16}:ISEGMent:FREQuency:FSTEp <NRf>
:SENSe{1-16}:ISEGMent:FREQuency:FSTEp?

Description: Sets the frequency step size in the index-based segment being defined. Outputs the frequency step size in the index-based segment being defined. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: 0 Hz to Maximum Instrument Frequency

Syntax Example: **:SENS:ISEGM:FREQ:FSTE 1.0E9**

:SENS:ISEGM:FREQ:FSTE?

:SENSe{1-16}:ISEGMent:FREQuency:FSTOp <NRf>
:SENSe{1-16}:ISEGMent:FREQuency:FSTOp?

Description: Sets the stop frequency in the index-based segment being defined. Outputs the stop frequency in the index-based segment being defined. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency.

Step Size = 0 Hz to Maximum Instrument Frequency.

Default Value: Maximum Instrument Frequency

Syntax Example: **:SENS:ISEGM:FREQ:FSTO 9.0E9**

:SENS:ISEGM:FREQ:FSTO?

:SENSe{1-16}:ISEGMent:FREQuency[:CW][:FIXed] <NRf>
:SENSe{1-16}:ISEGMent:FREQuency[:CW][:FIXed]?

Description: Sets the CW frequency in the indicated index-based segment. Outputs the CW frequency in the indicated index-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency

Default Value: 7.00000000000E+004

Syntax Example: **:SENS:ISEGM:FREQ 1.00E8**

:SENS:ISEGM:FREQ?

:SENSe{1-16}:ISEGMent:INDex:ACTive:STARt?

Description: Query only. Outputs the start index of the first active index-based segment. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NR1> The output parameter is an integer.

Range: 0 to 16,000/20,000

Default Value: 0

Syntax Example: **:SENS:ISEGM:IND:ACT:STAR?**

:SENSe{1-16}:ISEGMent:INDex:ACTive:STOP?

Description: Query only. Outputs the stop index of the last active index-based segment. The Maximum Instrument Points (MIP) setting is 16,001.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 16,000/20,000

Default Value: 14

Syntax Example: **:SENS:ISEGM:IND:ACT:STOP?**

:SENSe{1-16}:ISEGMent:INDex:STARt <NRf>
:SENSe{1-16}:ISEGMent:INDex:STARt?

Description: Sets the start index of the index-based segmented sweep. Outputs the start index of the index-based segmented sweep. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 16,000/20,000

Default Value: 0

Syntax Example: **:SENS:ISEGM:IND:STAR 0**

:SENS:ISEGM:IND:STAR?

:SENSe{1-16}:ISEGMent:INDex:STOP <NRf>
:SENSe{1-16}:ISEGMent:INDex:STOP?

Description: Sets the stop index of the index-based segmented sweep. Outputs the stop index of the index-based segmented sweep.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 14

Default Value: 14

Syntax Example: **:SENS:ISEGM:IND:STOP?**

:SENS:ISEGM:IND:STOP 1

:SENSe{1-16}:ISEGMent:MAXPoints?

Description: Query only. Outputs the total number of sweep points in the index-based segments. The number of ISEGMent total points can range from 1 to the Maximum Instrument Points. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to Maximum Instrument Points

Default Value: 15

Syntax Example: **:SENS:ISEGM:MAXP?**

:SENSe{1-16}:ISEGMent:POINt?

Description: Query only. Outputs the displayed number of sweep points in the index-based segments. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Query Parameters: <NR1> The output parameter is an integer.

Range: NA

Default Value: 15

Syntax Example: **:SENS:ISEGM:POIN?**

:SENSe{1-16}:ISEGMent:SPAntype <char>
:SENSe{1-16}:ISEGMent:SPAntype?

Description: Sets the span type of the index-based segment being defined as STARTSTOP or STARTSTEP. Outputs the span type as STARTSTOP or STARTSTEP of the index-based segment being defined.

STARTSTOP Selected

If STARTSTOP is selected, each segment is defined by the:

- Segment Start Frequency
- Segment Stop Frequency
- Number of Segment Points

STARTSTEP Selected

If STARTSTEP is selected, each segment is defined by the:

- Start Segment Frequency
- Frequency Step Size
- Number of Segment Points

Cmd Parameters: <char> STARTSTOP | STARTSTEP

Query Parameters: <char> STARTSTOP | STARTSTEP

Range: NA

Default Value: STARTSTOP

Syntax Example: **:SENS:ISEGM:SPA STARTSTOP**

:SENS:ISEGM:SPA?

:SENSe{1-16}:ISEGMent:SWEep:MAXimize

Description: Maximizes the index range of the index-based segmented sweep. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SENS:ISEGM:SWE:MAX**

:SENSe{1-16}:ISEGMent:SWEep:POINt <NR1>
:SENSe{1-16}:ISEGMent:SWEep:POINt?

Description: Sets the number of sweep points in the index-based segment being defined. Outputs the number of sweep points in the index-based segment being defined. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NR1> The input parameter is an integer.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 16,001/20,001

Default Value: 15

Syntax Example: **:SENS:ISEGM:SWE:POIN 15**

:SENS:ISEGM:SWE:POIN?

:SENSe{1-16}:ISEGMent[:STATe] <char>
:SENSe{1-16}:ISEGMent[:STATe]?

Description: Toggles the index-based segment being defined on and off. Outputs the on/off state of the index-based segment being defined.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 1

Syntax Example: **:SENS:ISEGM ON**

:SENS:ISEGM?

5-60 :SENSe{1-16}:ISEGMent{1-50} Subsystem

The :SENSe{1-16}:ISEGMent{1-50} subsystem commands are used to configure the indicated index-based segment. To configure only the active index-based segment, use:

- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.

Limit Line and Segment Subsystems

Related limit line and segment configuration and control subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107
- ":SENSe{1-16}:FSEGMent Subsystem" on page 5-211.
- ":SENSe{1-16}:FSEGMent{1-50} Subsystem" on page 5-218.
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231
- ":SENSe{1-16}:SEGMent Subsystem" on page 5-235

:SENSe{1-16}:ISEGMent{1-50}:CWMODe[:STATe] <char>
:SENSe{1-16}:ISEGMent{1-50}:CWMODe[:STATe]?

Description: Sets the Outputs and CW mode on/off state in the indicated index-based segment.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:SENS:ISEGM1:CWMOD ON**

:SENS:ISEGM1:CWMOD?

:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTArt <NRf>
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTArt?

Description: Sets the start frequency in the indicated index-based segment. The query outputs the start frequency in the indicated index-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency

Step Size = 0 Hz to Maximum Instrument Frequency

Default Value: Default Start Frequency (for normal sweeps, i.e. 300kHz here)

Syntax Example: **:SENS:ISEGM1:FREQ:FSTA 5.0E9**

:SENS:ISEGM1:FREQ:FSTA?

:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTEp <NRf>
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTEp?

Description: Sets the frequency step size in the indicated index-based segment. The query outputs the frequency step size in the indicated index-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Frequency Step Size to (Maximum Instrument Frequency minus Minimum Instrument Frequency)

Syntax Example: `:SENS:ISEGM1:FREQ:FSTE 1.00E4`

`:SENS:ISEGM1:FREQ:FSTE?`

:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTOp <NRf>
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTOp?

Description: Sets the stop frequency in the indicated index-based segment. Outputs the stop frequency in the indicated index-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Range = Minimum Instrument Frequency to Maximum Instrument Frequency. Step Size = 0 Hz to Maximum Instrument Frequency.

Default Value: Maximum Instrument Frequency

Syntax Example: `:SENS:ISEGM1:FREQ:FSTO 9.0E9`

`:SENS:ISEGM1:FREQ:FSTO?`

:SENSe{1-16}:ISEGMent{1-50}:FREQuency[:CW][:FIXed] <NRf>
:SENSe{1-16}:ISEGMent{1-50}:FREQuency[:CW][:FIXed]?

Description: Sets the start frequency in the indicated index-based segment. Outputs the start frequency in the indicated index-based segment. The Minimum Instrument Frequency (Fmin) depends on the instrument installed options. The Maximum Instrument Frequency (Fmax) depends on the instrument model. The Minimum Frequency Step Size is equal to 1 Hz.

See "Minimum/Maximum Instrument Frequency and Related Parameters" on page 1-13 for frequency limits for combinations of instrument model and available options.

Cmd Parameters: <NRf> The input parameter is in Hertz.

Query Parameters: <NR3> The output parameter is in Hertz.

Range: Minimum Instrument Frequency to Maximum Instrument Frequency

Default Value: Default Start Frequency

Syntax Example: **:SENS:ISEGM1:FREQ 2.0E9**

:SENS:ISEGM1:FREQ?

:SENSe{1-16}:ISEGMent{1-50}:SPAntype <char>
:SENSe{1-16}:ISEGMent{1-50}:SPAntype?

Description: Sets the span type of the indicated index-based segment. Outputs the span type of the indicated index-based segment.

STARTSTOP Selected

If STARTSTOP is selected, each segment is defined by the:

- Segment Start Frequency
- Segment Stop Frequency
- Number of Segment Points

STARTSTEP Selected

If STARTSTEP is selected, each segment is defined by the:

- Start Segment Frequency
- Frequency Step Size
- Number of Segment Points

Cmd Parameters: <char> STARTSTOP | STARTSTEP

Query Parameters: <char> STARTSTOP | STARTSTEP

Range: NA

Default Value: STARTSTOP

Syntax Example: **:SENS:ISEGM1:SPA STARTSTOP**

:SENS:ISEGM1:SPA?

:SENSe{1-16}:ISEGMent{1-50}:SWEep:POINt <NR1>
:SENSe{1-16}:ISEGMent{1-50}:SWEep:POINt?

Description: Sets the number of sweep points in the indicated index-based segment. Outputs the number of sweep points in the indicated index-based segment. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NR1> The input parameter is an integer.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to maximum number of instrument points depending on CW mode.

Default Value: 15

Syntax Example: **:SENS:ISEGM1:SWE:POIN 1.01E2**

:SENS:ISEGM1:SWE:POIN?

:SENSe{1-16}:ISEGMent{1-50}[:STATe] <char>
:SENSe{1-16}:ISEGMent{1-50}[:STATe]?

Description: Turns the indicated index-based segment on/off. Outputs the indicated index-based segment on/off state.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: 0

Syntax Example: **:SENS:ISEGM1**

:SENS:ISEGM1?

5-61 :SENSe{1-16}:RECEiver Subsystem

The :SENSe{1-16}:RECEiver subsystem command is used to the configure the VNA receiver.

:SENSe{1-16}:RECEiver:CONFiguration <char>
:SENSe{1-16}:RECEiver:CONFiguration?

Description: Sets the receiver configuration. Available command parameters depend on the instrument series and model number, where:

- STANdard = The receiver frequency follows the source frequency.

Outputs the receiver configuration.

Cmd Parameters: <char> STANdard

Query Parameters: <char> STAN

Range: NA

Default Value: NA

Syntax Example: **:SENS:RECE:CONF STAN**

:SENS:RECE:CONF?

5-62 :SENSe{1-16}:SEGMent Subsystem

The :SENSe{1-16}:SEGMent subsystem command is used to query the segmented sweep type as frequency-based or index based on the active sweep.

Limit Line and Segment Subsystems

Related limit line and segment configuration and control subsystems are:

- ":CALCulate{1-16}[:SELected]:LIMit Subsystem" on page 5-69
- ":DISPlay Subsystem" on page 5-107
- ":SENSe{1-16}:FSEGMent Subsystem" on page 5-211.
- ":SENSe{1-16}:FSEGMent{1-50} Subsystem" on page 5-218.
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231.
- ":SENSe{1-16}:ISEGMent{1-50} Subsystem" on page 5-231
- ":SENSe{1-16}:SEGMent Subsystem" on page 5-235

:SENSe{1-16}:SEGMent:TYPe?

Description: Query only. Outputs the segmented sweep type for the active segment as frequency-based or index-based where:

- FREQ = Frequency-based sweep type
- INDEX = Index-based sweep type

Query Parameters: <char> FREQ | INDEX

Range: NA

Default Value: NA

Syntax Example: **:SENS:SEGM:TYP?**

5-63 :SENSe{1-16}:SWEep Subsystem

The :SENSe{1-16}:SWEep subsystem commands are used to configure and control the instrument sweeps.

Sweep Subsystems

Related sweep configuration and control subsystems are:

- ":SENSe{1-16}:AVERage Subsystem" on page 5-136
- ":SENSe{1-16}:FREQuency Subsystem" on page 5-209
- ":SENSe{1-16}:SWEep Subsystem" on page 5-236

:SENSe{1-16}:SWEep:CW:POINt <NRf>
:SENSe{1-16}:SWEep:CW:POINt?

Description: Sets the CW sweep mode number of points. Outputs the CW sweep mode number of points.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 1 to 16,001/20,001

Default Value: 1

Syntax Example: **:SENS:SWE:CW:POIN 1.01E2**

:SENS:SWE:CW:POIN?

:SENSe{1-16}:SWEep:CW[:STATe] <char>
:SENSe{1-16}:SWEep:CW[:STATe]?

Description: Turns on/off the CW sweep mode. Outputs the on/off status of the CW sweep mode.

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default: 0

Syntax Example: **:SENS:SWE:CW 1**

:SENS:SWE:CW?

:SENSe{1-16}:SWEep:DELay <NRf>
:SENSe{1-16}:SWEep:DELay?

Description: Sets the sweep delay time. Outputs the sweep delay time.

Cmd Parameters: <NRf> The input parameter is in Seconds.

Query Parameters: <NR3> The output parameter is in Seconds.

Range: 0 to 100

Default Value: 0.00000000000E+000

Syntax Example: **:SENS:SWE:DEL 50.0E-3**

:SENS:SWE:DEL?

:SENSe{1-16}:SWEep:POINt <NRf>
:SENSe{1-16}:SWEep:POINt?

Description: Sets the number of measurement points. Outputs the number of measurement points. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 2 to 16,001/20,001 points.

Default Value: 201

Syntax Example: **:SENS:SWE:POIN 1.01E2**

:SENS:SWE:POIN?

:SENSe{1-16}:SWEep:TYPe <char>
:SENSe{1-16}:SWEep:TYPe?

Description: Sets the sweep type, where the available sweep types are:

- LINear = Frequency-based linear sweep
- LOGarithmic = Frequency-based logarithmic sweep
- FSEGMent = Segment-based sweep with frequency-based segments
- ISEGMent = Index-based sweep with frequency-based segments

Outputs the sweep type.

Cmd Parameters: <char> LINear | LOGarithmic|FSEGMent | ISEGMent

Query Parameters: <char> LIN | LOG|FSEGM | ISEGM

Range: NA

Default Value: LIN

Syntax Example: **:SENS:SWE:TYP LIN**

:SENS:SWE:TYP?

5-64 :SOURce{1-16}:POWer Subsystem

The :SOURce{1-16}:EFFective subsystem command is used to output the power level on the indicated port.

Power Configuration Subsystems

Related power configuration and control systems are:

- ":SOURce{1-16}:POWer Subsystem" on page 5-238
- ":STATus:OPERation Subsystem" on page 5-239

:SOURce{1-16}:POWer <char>

Description: Query only. Outputs the effective power level of the given port on the given channel.

Cmd parameters: <char> HIGH | LOW

Query Parameters: <NR3> The output parameter is in dBm.

Range: –3 and –20 dBm

Default Value: –3 dBm

Syntax Example: `:SOUR1:POW?`

5-65 :STATus:OPERation Subsystem

The :STATus:OPERation subsystem commands are used to set and output values from the Operation Status Enable Register (refer to Figure 2-3, "Status Register Structure" on page 2-29).

:STATus:OPERation:CONDition?

Description: Query only. Outputs the value of the operation status condition register.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 32767

Default Value: 0

Syntax Example: **:STAT:OPER:COND?**

:STATus:OPERation:ENABle <NRf>
:STATus:OPERation:ENABle?

Description: Sets the value of the operation status enable register. Outputs the value of the operation status enable register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:OPER:ENAB 1**

:STAT:OPER:ENAB?

:STATus:OPERation:NTRansition <NRf>
:STATus:OPERation:NTRansition?

Description: Sets the value of the negative transition filter of the operation status register. Outputs the value of the negative transition filter of the operation status register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:OPER:NTR 1**

:STAT:OPER:NTR?

:STATus:OPERation:PTRansition <NRf>
:STATus:OPERation:PTRansition?

Description: Sets the value of the positive transition filter of the operation status register. Outputs the value of the positive transition filter of the operation status register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:OPER:PTR 1**

:STAT:OPER:PTR?

:STATus:OPERation[:EVENt]?

Description: Query only. Outputs the value of the operation status event register.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 32767

Default Value: 0

Syntax Example: **:STAT:OPER?**

5-66 :STATus:QUEStionable Subsystem

The :STATus:QUEStionable subsystem commands are used to set and output values from the Questionable Status Enable Register.

:STATus:QUEStionable:CONDition?

Description: Query only. Outputs the value of the questionable status condition register.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 32767

Default Value: 0

Syntax Example: **:STAT:QUES:COND?**

:STATus:QUEStionable:ENABle <NRf>
:STATus:QUEStionable:ENABle?

Description: Sets the value of the questionable status enable register. Outputs the value of the questionable status enable register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:QUES:ENAB 1**

:STAT:QUES:ENAB?

:STATus:QUEStionable:LIMit:CONDition?

Description: Query only. Outputs the value of the questionable limit status condition register.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 32767

Default Value: 0

Syntax Example: **:STAT:QUES:LIM:COND?**

:STATus:QUEStionable:LIMit:ENABle <NRf>
:STATus:QUEStionable:LIMit:ENABle?

Description: Sets the value of the questionable limit status enable register. Outputs the value of the questionable limit status enable register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:QUES:LIM:ENAB 1**

:STAT:QUES:LIM:ENAB?

:STATus:QUEStionable:LIMit:NTRansition <NRf>
:STATus:QUEStionable:LIMit:NTRansition?

Description: Sets the value of the negative transition filter of the questionable limit status register. Outputs the value of the negative transition filter of the questionable limit status register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:QUES:LIM:NTR 1**

:STAT:QUES:LIM:NTR?

:STATus:QUEStionable:LIMit:PTRansition <NRf>
:STATus:QUEStionable:LIMit:PTRansition?

Description: Sets the value of the positive transition filter of the questionable limit status register. Outputs the value of the positive transition filter of the questionable limit status register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: 0

Syntax Example: **:STAT:QUES:LIM:PTR 1**

:STAT:QUES:LIM:PTR?

:STATus:QUEStionable:LIMit[:EVENt]?

Description: Query only. Outputs the value of the questionable limit status event register.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 32767

Default Value: 0

Syntax Example: **:STAT:QUES:LIM?**

:STATus:QUEStionable:NTRansition <NRf>
:STATus:QUEStionable:NTRansition?

Description: Sets the value of the negative transition filter of the questionable status register. Outputs the value of the negative transition filter of the questionable status register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:QUES:NTR 1**

:STAT:QUES:NTR?

:STATus:QUEStionable:PTRansition <NRf>
:STATus:QUEStionable:PTRansition?

Description: Sets the value of the positive transition filter of the questionable status register. Outputs the value of the positive transition filter of the questionable status register.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 65535

Default Value: NA

Syntax Example: **:STAT:QUES:PTR 1**

:STAT:QUES:PTR?

:STATus:QUEStionable[:EVENt]?

Description: Query only. Outputs the value of the questionable status event register.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 32767

Default Value: 0

Syntax Example: **:STAT:QUES?**

5-67 :SYSTem Subsystem

The :SYSTEM subsystem commands configure and control various system-level instrument settings.

- :SYSTem:COMMunicate commands are used to configure and control network communications.
- :SYSTem:ERRor commands are used to query and clear the contents of the Error Queue.
- :SYSTem:IFCalibration commands set the system IF calibration.
- :SYSTem:POINt sets the maximum number of measurement points, and re-boots the instrument is required.
- :SYSTem:PORT returns the number of ports on the instrument.
- :SYSTem:PRESet commands configure the instrument preset/reset configuration, and if required, set the preset/reset configuration file.

:SYSTem:COMMunicate:TCPIP:ADDRess?

Description: Query only. Outputs the IP address of the Ethernet interface. The setting cannot be changed by the user.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: 0.0.0.0 to 255.255.255.255

Default Value: Varies with installation.

Syntax Example: **:SYST:COMM:TCPIP:ADDR?**

:SYSTem:COMMunicate:TCPIP:GATE?

Description: Query only. Outputs the default Gateway of the Ethernet interface. The setting cannot be changed by the user.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: 0.0.0.0 to 255.255.255.255

Default Value: Varies with installation.

Syntax Example: **:SYST:COMM:TCPIP:GATE?**

:SYSTem:COMMunicate:TCPIP:HDW?

Description: Query only. Outputs the MAC hardware address of the Ethernet interface. The setting cannot be changed by the user.

Query Parameters: <char> The output parameter can be any combination of numbers and letters.

Range: NA

Default Value: Varies with individual instrument.

Syntax Example: **:SYST:COMM:TCPIP:HDW?**

:SYSTem:COMMunicate:TCPIP:MASK?

Description: Query only. Outputs the instrument TCP/IP port address. The setting cannot be changed by the user.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0.0.0.0 to 255.255.255.255

Default Value: Varies with installation.

Syntax Example: **:SYST:COMM:TCPIP:MASK?**

:SYSTem:COMMunicate:TCPIP:PORT <NRf>
:SYSTem:COMMunicate:TCPIP:PORT?

Description: Enters the instrument TCP/IP port address. This value is user definable. The recommended TCP/IP address should be greater than or equal to 5001. Outputs the instrument TCP/IP port address.

Cmd Parameters: <NRf> The input parameter is a unitless number.

Query Parameters: <NR1> The output parameter is an integer.

Range: MPNI

Default Value: 5001

Syntax Example: **:SYST:COMM:TCPIP:PORT 5001**

:SYST:COMM:TCPIP:PORT?

:SYSTem:ERRor:CLEar

Description: Clears the contents of the error queue. No query.

Cmd Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SYST:ERR:CLE**

:SYSTem:ERRor:COUNt?

Description: Query only. Outputs the number of errors in the error queue.

Query Parameters: <NR1> The output parameter is an integer.

Range: 0 to 100

Default Value: 0

Syntax Example: **:SYST:ERR:COUN?**

:SYSTem:ERRor:QUEue?

Description: Query only. Outputs the contents of the error queue.

Query Parameters: <block> See definition of "<block> or <arbitrary block>" on page 2-10.

Range: NA

Default Value: NA

Syntax Example: **:SYST:ERR:QUE?**

:SYSTem:ERRor[:NEXt]?

Description: Query only. Removes and outputs the oldest error in the error queue.

Query Parameters: <ASCII> See definition of "<ASCII> or <Arbitrary ASCII>" on page 2-10.

Range: NA

Default Value: No Error

Syntax Example: **:SYST:ERR?**

:SYSTem:FACTorycal:DELete

Description: Deletes the factory calibration

Range: NA

Default Value: NA

Syntax Example: `:SYST:FACT:DEL`

:SYSTem:FACTorycal:RECover

Description: Recovers the factory calibration

Range: NA

Default Value: NA

Syntax Example: `:SYST:FACT:REC`

:SYSTem:HOLD:RF[:STATe] <char>
:SYSTem:HOLD:RF[:STATe]?

Description: Sets the RF on/off state in Hold. Outputs the RF on/off state in Hold

Cmd Parameters: <char> 1 | 0 | ON | OFF

Query Parameters: <char> 1 | 0

Range: NA

Default Value: NA

Syntax Example: `:SYST:HOLD:RF ON`

`:SYST:HOLD:RF?`

:SYSTem:IFCalibration:TRIGger

Description: Triggers an IF calibration. No query.

Cmd Parameters: NA

Syntax Example: **`:SYST:IFC:TRIG`**

:SYSTem:POINt:MAXimum?

Description: The query returns the maximum number of points the instrument can measure in a sweep. For MS46122A/B and MS46322A/B Series VNAs, the Maximum Instrument Points (MIP) is 16,001. For the MS46121A/B, the Maximum Instrument Points (MIP) is 20,001.

Query Parameters: <char> 16001/20001

Range: NA

Default Value: 16001/20001

Syntax Example: **`:SYST:POIN:MAX?`**

:SYSTem:PORT:COUNt?

Description: Query only. The query outputs the number of instrument test ports.

Cmd Parameters: NA

Query Parameters: 2

Range: NA

Default Value: 2

Syntax Example: `:SYST:PORT:COUN?`

:SYSTem:POWerup:FILe <string>
:SYSTem:POWerup:FILe?

Description: The command sets the file path for the .cha power up configuration file to use on power up. The query outputs the file path of the .cha used for power up configuration.

Cmd Parameters: <string> Filename and path in the form: 'x:\directory\filename.cha' where x:\directory\filename.cha must exist. See definition of "<string>" on page 2-10.

Query Parameters: <char> Filename and path in the form: x:\directory\filename.cha

Range: NA

Default Value: NA

Syntax Example:
```
:SYST:POW:FIL 'C:\filepath\filename.cha'
:SYST:POW:FIL?
```

:SYSTem:POWerup:TYPe <char>
:SYSTem:POWerup:TYPe?

Description: The command sets the power up instrument state to reset, last, or user defined. If USER is selected, a previously saved user-defined power up configuration file must exist and be stored on the instrument hard disk drive. The query outputs the power up instrument state as reset, last, or user defined.

Cmd Parameters: <char> RESET | LAST | USER

Query Parameters: <char> RESET | LAST | USER

Range: NA

Default Value: LAST

Syntax Example:
```
:SYST:POW:TYP RESET
:SYST:POW:TYP?
```

:SYSTem:PRESet

Description: Performs an instrument preset. The instrument returns to its factory as-shipped configuration, typically displaying four traces set to:

- Trace 1 (Tr1) set to Log Magnitude.
- Trace 2 (Tr2) set to Log Magnitude.
- Trace 3 (Tr3) set to Log Magnitude.
- Trace 4 (Tr4) set to Log Magnitude.

- Clears any user-defined segmented limit lines.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SYST:PRES**

:SYSTem:PRESet:FILe <string>
:SYSTem:PRESet:FILe?

Description: The commands sets the file path for the .cha file to use on preset. The query outputs the file path for the CHA file used for preset.

Cmd Parameters: <string> The directory and filename in the form: 'x:\directory\filename.cha' where x:\directory\filename.cha must exist. See definition of "<string>" on page 2-10.

Query Parameters: <char> Filename and path in the form: x:\directory\filename.cha

Range: NA

Default Value: NA

Syntax Example:
```
:SYST:PRES:FIL 'C:\directory\filename.cha'
:SYST:PRES:FIL?
```

:SYSTem:PRESet:TYPe <char>
:SYSTem:PRESet:TYPe?

Description: The command sets the preset type to RESET, RESET0 (reset zero), or USER. The query outputs the selected preset type. The different preset types provide the following functions:

- USER - If the USER type is selected, a previously saved user-defined preset file must exist on the instrument disk drive. Once set, issuing a :SYSTem:PRESet command returns the instrument to the user-defined configuration. The user-defined preset file name is set using the :SYSTem:PRESet:FILe command.
- RESET - If the RESET type is selected, issuing a :SYSTem:PRESet command returns the instrument to the factory as-shipped configuration.
- RESET0 - If the RESET0 type is selected, issuing the :SYSTem:PRESet command clears the instrument memory of all settings, configurations and returns the instrument to a factory as-shipped state. Functionally, issuing a :SYSTem:PRESet:TYPe RESET0 command followed by a :SYSTem:PRESet command is the same as issuing a :SYSTem:PRESet:ZERo command. Note that the RESET0 parameter does not delete any files. See the :SYSTem:PRESet:ZERo command description for a complete listing of included and excluded preset elements.

The preset type can also be configured by from the front panel by:

- Navigating to MAIN | System | SYSTEM | Setup | SETUP | Preset Setup | PRESET SETUP
- On the PRESET SETUP menu, selecting either Default, Default 0, or Saved Setup.
- If Saved Setup is to be used, the user-defined configuration file must be saved to the instrument solid-state drive. Then select the Select Saved Setup File button and select the desired user-defined preset file. Then select Saved Setup.

A preset can also be executed from the front panel by doing one of the following actions:

- Pressing the front panel **Preset key**.
- Selecting the ICON TOOLBAR | Preset icon.
- Selecting the MENU BAR | Utilities | Preset command.

Cmd Parameters: <char> RESET | RESET0 | USER

Query Parameters: <char> RESET | RESET0 | USER

Range: NA

Default Value: RESET

Syntax Example: `:SYST:PRES:TYP RESET`

`:SYST:PRES:TYP?`

:SYSTem:PRESet:ZERo

Description: Performs an instrument preset (same as `:SYST:PRES`). The instrument returns to its factory as-shipped configuration, typically displaying four traces set to:

- Trace 1 (Tr1) set to Log Magnitude.
- Trace 2 (Tr2) set to Log Magnitude.
- Trace 3 (Tr3) set to Log Magnitude.
- Trace 4 (Tr4) set to Log Magnitude.
- Clears any user-defined segmented limit lines.
- Clears any user-defined calibration kits.

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **:SYST:PRES:ZER**

:TRIGger[:SEQuence]:SOURce <char>
:TRIGger[:SEQuence]:SOURce?

Description: This command sets the source of sweep/measurement triggering where the following options are available:

- INTernal = The triggering source is internal.
- EXTernal = The triggering source is from the rear panel external trigger port.

The query outputs the source of the sweep/measurement triggering.

Cmd Parameters: <char> INTernal | EXTernal

Query Parameters: <char> INT | EXT

Range: NA

Default Value: INT

Syntax Example: **:TRIG:SOUR INT**

:TRIG:SOUR?

:TRIGger[:SEQuence][:IMMediate][:REMote]

Description: Triggers a continuous sweep from the remote interface. During the sweep, command execution continues and does not pause. The operation of this command depends on the settings of the :SENSe{1-16}:HOLD:FUNCtion, :TRIGger[:SEQuence][:REMote]:SINGle, and :TRIGger[:SEQuence][:IMMediate][:REMote] commands, described below. No query.

See the following subsystems for additional information:

- ":SENSe{1-16}:ABORtcal Subsystem" on page 5-136
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225

:SENSe{1-16}:HOLD:FUNC CONT and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion CONTinuous

// Sweep State = The sweep is sweeping continuously

// Command Execution = The parser is ready for a command right
away.

:TRIGger[:SEQuence][:IMMediate][:REMote]

// Sweep State = The sweep restarts and sweeps continuously. When
the sweep gets to the end of the sweep, it continues to sweep.
There is NO STATUS information that the end of the sweep has been
reached.

// Command Execution = The parser is ready for a command right away
```

:SENSe{1-16}:HOLD:FUNC CONT and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion CONTinuous
// Sweep State = The sweep is sweeping continuously.
// Command Execution = The parser is ready for a command right away
:TRIGger[:SEQuence][:REMote]:SINGle
// Sweep State = The sweep restarts and sweeps continuously. When
the sweep gets to the end of the sweep, it sets the end of sweep
status bit and continues to sweep.
// Command Execution = Further execution is blocked until the end
of the sweep.
// Command Execution resumes when the sweep has reached the end of
the sweep.
```

:SENSe{1-16}:HOLD:FUNC HOLD and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion HOLD
// Sweep State = The sweep is stopped.
// Command Execution = The parser is ready for a command right away
:TRIGger[:SEQuence][:IMMediate][:REMote]
// Sweep State = The command has no effect. The sweep is stopped.
// Command Execution = The parser is ready for a command right away
```

:SENSe{1-16}:HOLD:FUNC HOLD and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion HOLD
// Sweep State = The sweep is stopped
// Command Execution = The parser is ready for a command right
away.
:TRIGger[:SEQuence][:REMote]:SINGle
// Sweep State = The sweep restarts and sweeps until the end of the
sweep, at which point it sets the end of sweep status bit and
stops.
// Command Execution = Further execution is blocked until the end
of the sweep.
// Command Execution resumes when the sweep has reached the end of
the sweep.
```

:SENSe{1-16}:HOLD:FUNC SING and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion SINGle
// Sweep State = The sweep does one complete sweep, goes into hold
and stops.
// Command Execution = The parser is ready for a command.
:TRIGger[:SEQuence][:IMMediate][:REMote]
// Sweep State = The command has no effect. The sweep is stopped.
// Command Execution = The parser is ready for a command.
```

:SENSe{1-16}:HOLD:FUNC SING and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion SINGle

// Sweep State = The sweep does one complete sweep, goes into hold
and stops.

// Command Execution = The parser is ready for a command right
away.

:TRIGger[:SEQuence][:REMote]:SINGle

// Sweep State = The sweep restarts and sweeps until the end of the
sweep, at which point it sets the end of sweep status bit and
stops.

// Command Execution = Further execution is blocked until the end
of the sweep.

// Command Execution resumes when the sweep has reached the end of
the sweep.
```

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:TRIG`**

:TRIGger[:SEQuence][:REMote]:SINGle

Description: Triggers a single sweep with synchronization from the remote interface. During the sweep, command execution pauses until the sweep is complete. The operation of this command is modified by the instrument state set by the :SENSe{1-16}:HOLD:FUNCtion command. No query.

See the following subsystems for additional information:

- ":SENSe{1-16}:ABORtcal Subsystem" on page 5-136
- ":SENSe{1-16}:ISEGMent Subsystem" on page 5-225

Variable Operation

The operation of this command depends on the settings of the :SENSe{1-16}:HOLD:FUNCtion, :TRIGger[:SEQuence][:REMote]:SINGle, and :TRIGger[:SEQuence][:IMMediate][:REMote] commands. Each setting combination is described in the sections below.

:SENSe{1-16}:HOLD:FUNC CONT and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion CONTinuous

// Sweep State = The sweep is sweeping continuously

// Command Execution = The parser is ready for a command right
away.

:TRIGger[:SEQuence][:IMMediate][:REMote]

// Sweep State = The sweep restarts and sweeps continuously. When
the sweep gets to the end of the sweep, it continues to sweep.
There is NO STATUS information that the end of the sweep has been
reached.

// Command Execution = The parser is ready for a command right away
```

:SENSe{1-16}:HOLD:FUNC CONT and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion CONTinuous
// Sweep State = The sweep is sweeping continuously.
// Command Execution = The parser is ready for a command right away
:TRIGger[:SEQuence][:REMote]:SINGle
// Sweep State = The sweep restarts and sweeps continuously. When
the sweep gets to the end of the sweep, it sets the end of sweep
status bit and continues to sweep.
// Command Execution = Further execution is blocked until the end
of the sweep.
// Command Execution resumes when the sweep has reached the end of
the sweep.
```

:SENSe{1-16}:HOLD:FUNC HOLD and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion HOLD
// Sweep State = The sweep is stopped.
// Command Execution = The parser is ready for a command right away
:TRIGger[:SEQuence][:IMMediate][:REMote]
// Sweep State = The command has no effect. The sweep is stopped.
// Command Execution = The parser is ready for a command right away
```

:SENSe{1-16}:HOLD:FUNC HOLD and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion HOLD
// Sweep State = The sweep is stopped
// Command Execution = The parser is ready for a command right
away.
:TRIGger[:SEQuence][:REMote]:SINGle
// Sweep State = The sweep restarts and sweeps until the end of the
sweep, at which point it sets the end of sweep status bit and
stops.
// Command Execution = Further execution is blocked until the end
of the sweep.
// Command Execution resumes when the sweep has reached the end of
the sweep.
```

:SENSe{1-16}:HOLD:FUNC SING and :TRIG

```
:SENSe{1-16}:HOLD:FUNCtion SINGle
// Sweep State = The sweep does one complete sweep, goes into hold
and stops.
// Command Execution = The parser is ready for a command.
:TRIGger[:SEQuence][:IMMediate][:REMote]
// Sweep State = The command has no effect. The sweep is stopped.
// Command Execution = The parser is ready for a command.
```

:SENSe{1-16}:HOLD:FUNC SING and :TRIG:SING

```
:SENSe{1-16}:HOLD:FUNCtion SINGle

// Sweep State = The sweep does one complete sweep, goes into hold
and stops.

// Command Execution = The parser is ready for a command right
away.

:TRIGger[:SEQuence][:REMote]:SINGle

// Sweep State = The sweep restarts and sweeps until the end of the
sweep, at which point it sets the end of sweep status bit and
stops.

// Command Execution = Further execution is blocked until the end
of the sweep.

// Command Execution resumes when the sweep has reached the end of
the sweep.
```

Cmd Parameters: NA

Query Parameters: NA

Range: NA

Default Value: NA

Syntax Example: **`:TRIG:SING`**

Appendix A — Agilent ENA SCPI Programming Emulation

A-1 Introduction

This chapter provides a list of Agilent ENA programming commands supported through software emulation. Agilent ENA commands are only available in Agilent mode, which is activated through the use of the command: `:CONFigure:BASe:MODe AGILent`. Any commands that fall in both the ENA and ShockLine command bases will be interpreted according to the current SCPI mode. To return to Anritsu mode, use the command `:CONFigure:BASe:MODe ANRitsu`.

For more detailed information about using these commands, refer to the appropriate ENA Programming Manual. Refer to Chapter 2, "Programming the ShockLine™ Series VNA" for definitions of parameters and other notations, and for a summary of command notational conventions used throughout this manual. For help with transitioning existing test software from Agilent ENA to Anritsu ShockLine VNAs, please contact Anritsu at ShockLineVNA.support@anritsu.com.

A-2 ENA Command Listing

```
:CALCulate{1-160}:PARameter{1-16}:DEFine<string>{S11|S21|S31|S41|S12|S22|S32|S42|S13
 |S23|S33|S43|S14|S24|S34|S44|A|B|C|D|R1|R2|R3|R4|Aux1|AUX2}
:CALCulate{1-160}:PARameter{1|-16}:DEFine?
:CALCulate{1-160}:PARameter{1-16}:SELect
:CALCulate{1-160}:PARameter:COUNt <numeric>
:CALCulate{1-160}:PARameter:COUNt?
:CALCulate{1-160}[:SELected]:DATA:FDATa <numeric1>,… ,<numericNOP*2>
:CALCulate{1-160}[:SELected]:DATA:FDATa?
:CALCulate{1-160}:TRACe{1-16}:DATA:FDATa<numeric1>,… ,<numeric NOP*2>
:CALCulate{1-160}:TRACe{1-16}:DATA:FDATa?
:CALCulate{1-160}[:SELected]:DATA:FMEMory <numeric1>,… ,<numeric NOP*2>
:CALCulate{1-160}[:SELected]:DATA:FMEMory?
:CALCulate{1-160}:TRACe{1-16}:DATA:FMEMory<numeric1>,… ,<numeric NOP*2>
:CALCulate{1-160}:TRACe{1-16}:DATA:FMEMory?
:CALCulate{1-160}[:SELected]:DATA:SDATa <numeric 1>,… ,<numericNOP*2>
:CALCulate{1-160}[:SELected]:DATA:SDATa?
:CALCulate{1-160}:TRACe{1-16}:DATA:SDATa<numeric1>,… ,<numeric NOP*2>
:CALCulate{1-160}:TRACe{1-16}:DATA:SDATa?
:CALCulate{1-160}[:SELected]:DATA:SMEMory <numeric1>,… ,<numeric NOP*2>
:CALCulate{1-160}[:SELected]:DATA:SMEMory?
:CALCulate{1-160}:TRACe{1-16}:DATA:SMEMory<numeric1>,… ,<numeric NOP*2>
:CALCulate{1-160}:TRACe{1-16}:DATA:SMEMory?
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:CENTer <numeric>
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:CENTer?
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:CENTer<numeric>
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:CENTer?
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:SPAN <numeric>
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:SPAN?
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:SPAN<numeric>
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:SPAN?
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:STARt <numeric>
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:STARt?
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:STARt<numeric>
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:STARt?
```

```
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:STATe{ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:STATe?
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:STATe{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:STATe?
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:STOP <numeric>
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME:STOP?
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:STOP<numeric>
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME:STOP?
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME[:TYPE]{BPASs|NOTCh}
:CALCulate{1-160}[:SELected]:FILTer[:GATE]:TIME[:TYPE]?
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME[:TYPE]{BPASs|NOTCh}
:CALCulate{1-160}:TRACe{1-16}:FILTer[:GATE]:TIME[:TYPE]?
:CALCulate{1-160}[:SELected]:LIMit:DISPlay[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:LIMit:DISPlay[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:LIMit:DISPlay[:STATe]{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:LIMit:DISPlay[:STATe]?
:CALCulate{1-160}[:SELected]:LIMit:FAIL?
:CALCulate{1-160}:TRACe{1-16}:LIMit:FAIL?
:CALCulate{1-160}[:SELected]:LIMit:REPort[:DATA]?
:CALCulate{1-160}:TRACe{1-16}:LIMit:REPort[:DATA]?
:CALCulate{1-160}[:SELected]:LIMit:REPort:POINts?
:CALCulate{1-160}:TRACe{1-16}:LIMit:REPort:POINts?
:CALCulate{1-160}[:SELected]:LIMit[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:LIMit[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:LIMit[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:LIMit[:STATe]?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:ACTivate
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:ACTivate
:CALCulate{1-160}[:SELected]:MARKer:COUPle {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer:COUPle?
:CALCulate{1-160}:TRACe{1-16}:MARKer:COUPle {ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MARKer:COUPle?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:DISCrete{ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer{1-10}:DISCrete?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:DISCrete{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:DISCrete?
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain:COUPle{ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain:COUPle?
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain:COUPle{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain:COUPle?
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain:STARt<numeric>
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain:STARt?
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain:STARt<numeric>
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain:STARt?
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain[:STATe]{ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain[:STATe]{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain[:STATe]?
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain:STOP<numeric>
:CALCulate{1-160}[:SELected]:MARKer:FUNCtion:DOMain:STOP?
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain:STOP<numeric>
:CALCulate{1-160}:TRACe{1-16}:MARKer:FUNCtion:DOMain:STOP?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion: PEXCursion<numeric>
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion: PEXCursion?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:PEXCursion <numeric>
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:PEXCursion?
```

```
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:PPOLarity{POSitive|NEGative|BOTH}
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:PPOLarity?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:PPOLarity
{POSitive|NEGative|BOTH}
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:PPOLarity?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:TARGet<numeric>
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:TARGet?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:TARGet<numeric>
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:TARGet?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:TRACking{ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:TRACking?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:TRACking {ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:TRACking?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:TTRansition{POSitive
 |NEGative|BOTH}
:CALCulate{1-160}[:SELected]:MARKer{1-10}:FUNCtion:TTRansition?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:TTRansition {POSitive
 |NEGative|BOTH}
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:FUNCtion:TTRansition?
:CALCulate{1-160}[:SELected]:MARKer{1-10}[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer{1-10}[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}[:STATe]{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}[:STATe]?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:X <numeric>
:CALCulate{1-160}[:SELected]:MARKer{1-10}:X?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:X <numeric>
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:X?
:CALCulate{1-160}[:SELected]:MARKer{1-10}:Y?
:CALCulate{1-160}:TRACe{1-16}:MARKer{1-10}:Y?
:CALCulate{1-160}[:SELected]:MARKer:REFerence[:STATe]{ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MARKer:REFerence[:STATe]?
:CALCulate{1-160}:TRACe(Tr):MARKer:REFerence[:STATe]{ON|OFF|1|0}
:CALCulate{1-160}:TRACe(Tr):MARKer:REFerence[:STATe]?
:CALCulate{1-160}[:SELected]:MATH:FUNCtion {NORMal|SUBTract|DIVide|ADD|MULTiply}
:CALCulate{1-160}[:SELected]:MATH:FUNCtion?
:CALCulate{1-160}:TRACe{1-16}:MATH:FUNCtion {NORMal|SUBTract|DIVide|ADD|MULTiply}
:CALCulate{1-160}:TRACe{1-16}:MATH:FUNCtion?
:CALCulate{1-160}[:SELected]:MATH:MEMorize
:CALCulate{1-160}:TRACe{1-16}:MATH:MEMorize
:CALCulate{1-160}[:SELected]:MSTatistics:DATA?
:CALCulate{1-160}:TRACe{1-16}:MSTatistics:DATA?
:CALCulate{1-160}[:SELected]:MSTatistics[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:MSTatistics[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:MSTatistics[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:MSTatistics[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:SMOothing:APERture <numeric>
:CALCulate{1-160}:TRACe{1-16}:SMOothing:APERture?
:CALCulate{1-160}[:SELected]:SMOothing[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:SMOothing[:STATe]?
:CALCulate{1-160}:TRACe{1-16}:SMOothing[:STATe] {ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:SMOothing[:STATe]?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:CENTer <numeric>
:CALCulate{1-160}[:SELected]:TRANsform:TIME:CENTer?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:CENTer<numeric>
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:CENTer?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:IMPulse:WIDTh?
```

```
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:IMPulse:WIDTh?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:KBESsel <numeric>
:CALCulate{1-160}[:SELected]:TRANsform:TIME:KBESsel?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:KBESsel<numeric>
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:KBESsel?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:SPAN <numeric>
:CALCulate{1-160}[:SELected]:TRANsform:TIME:SPAN?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:SPAN <numeric>
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:SPAN?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STARt <numeric>
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STARt?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STARt <numeric>
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STARt?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STATe {ON|OFF|1|0}
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STATe?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STATe{ON|OFF|1|0}
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STATe?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STIMulus{IMPulse|STEP}
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STIMulus?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STIMulus{IMPulse|STEP}
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STIMulus?
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STOP <numeric>
:CALCulate{1-160}[:SELected]:TRANsform:TIME:STOP?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STOP <numeric>
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME:STOP?
:CALCulate{1-160}[:SELected]:TRANsform:TIME[:TYPE] {BPASs|LPASs}
:CALCulate{1-160}[:SELected]:TRANsform:TIME[:TYPE]?
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME[:TYPE]{BPASs|LPASs}
:CALCulate{1-160}:TRACe{1-16}:TRANsform:TIME[:TYPE]?
:DISPlay:ANNotation:FREQuency[:STATe] {ON|OFF|1|0}
:DISPlay:ANNotation:FREQuency[:STATe]?
:DISPlay:COLor{1|2}:BACK <numeric 1>,<numeric 2>,<numeric 3>
:DISPlay:COLor{1|2}:BACK?
:DISPlay:COLor{1|2}:GRATicule{1|2} <numeric 1>,<numeric2>,<numeric 3>
:DISPlay:COLor{1|2}:GRATicule{1|2}?
:DISPlay:COLor{1|2}:LIMit{1|2} <numeric 1>,<numeric 2>,<numeric3>
:DISPlay:COLor{1|2}:LIMit{1|2}?
:DISPlay:COLor{1|2}:RESet
:DISPlay:FSIGn {ON|OFF|1|0}
:DISPlay:FSIGn?
:DISPlay:IMAGe {NORMal|INVert}
:DISPlay:IMAGe?
:DISPlay:SKEY[:STATe] {ON|OFF|1|0}
:DISPlay:SKEY[:STATe]?
:DISPlay:WINDow{1-160}:MAXimize {ON|OFF|1|0}
:DISPlay:WINDow{1-160}:MAXimize?
:DISPlay:WINDow{1-160}:TITLe:DATA <string>
:DISPlay:WINDow{1-160}:TITLe:DATA?
:DISPlay:WINDow{1-160}:TITLe:STATe {ON|OFF|1|0}
:DISPlay:WINDow{1-160}:TITLe:STATe?
:DISPlay:WINDow{1-160}:TRACe{1-16} :MEMory[:STATe]{ON|OFF|1|0}
:DISPlay:WINDow{1-160}:TRACe{1-16} :MEMory[:STATe]?
:DISPlay:WINDow{1-160}:TRACe{1-16} :STATe {ON|OFF|1|0}
:DISPlay:WINDow{1-160}:TRACe{1-16} :STATe?
:DISPlay:WINDow{1-160}:TRACe{1-16} :Y[:SCALe]:PDIVision<numeric>
:DISPlay:WINDow{1-160}:TRACe{1-16} :Y[:SCALe]:PDIVision?
```

```
:DISPlay:WINDow{1-160}:TRACe{1-16} :Y[:SCALe]:RLEVel<numeric>
:DISPlay:WINDow{1-160}:TRACe{1-16} :Y[:SCALe]:RLEVel?
:DISPlay:WINDow{1-160}:TRACe{1-36} :Y[:SCALe]:RPOSition<numeric>
:DISPlay:WINDow{1-160}:TRACe{1-36} :Y[:SCALe]:RPOSition?
:FORMat:BORDer {NORMal|SWAPped}
:FORMat:BORDer?
:FORMat:DATA {ASCii|REAL|REAL32}
:FORMat:DATA?
:HCOPy:IMAGe {NORMal|INVert}
*CLS
*ESE <numeric>
*ESE?
*ESR?
*IDN?
*OPC
*OPT?
*RST
*SRE <numeric>
*SRE?
*STB?
*TRG
*WAI
:INITiate{1-160}:CONTinuous {ON|OFF|1|0}
:INITiate{1-160}:CONTinuous?
:MMEMory:CATalog? <string 1>
:MMEMory:COPY <string 1>,<string 2>
:MMEMory:DELete <string>
:MMEMory:LOAD:CKIT{1-30} <string>
:MMEMory:LOAD:LIMit <string>
:MMEMory:MDIRectory <string>
:MMEMory:STORe:IMAGe <string>
:MMEMory:STORe:LIMit <string>
:SENSe{1-160}:AVERage:CLEar
:SENSe{1-160}:AVERage:COUNt <numeric>
:SENSe{1-160}:AVERage:COUNt?
:SENSe{1-160}:AVERage[:STATe] {ON|OFF|1|0}
:SENSe{1-160}:AVERage[:STATe]?
:SENSe{1-160}:BANDwidth[:RESolution] <numeric>
:SENSe{1-160}:BANDwidth[:RESolution]?
:SENSe{1-160}:BWIDth[:RESolution] <numeric>
:SENSe{1-160}:BWIDth[:RESolution]?
:SENSe{1-160}:CORRection:STATe {ON|OFF|1|0}
:SENSe{1-160}:CORRection:STATe?
:SENSe{1-160}:FREQuency:CENTer <numeric>
:SENSe{1-160}:FREQuency:CENTer?
:SENSe{1-160}:FREQuency[:CW] <numeric>
:SENSe{1-160}:FREQuency[:CW]?
:SENSe{1-160}:FREQuency:FIXed <numeric>
:SENSe{1-160}:FREQuency:FIXed?
:SENSe{1-160}:FREQuency:DATA?
:SENSe{1-160}:FREQuency:SPAN <numeric>
:SENSe{1-160}:FREQuency:SPAN?
:SENSe{1-160}:FREQuency:STARt <numeric>
:SENSe{1-160}:FREQuency:STARt?
:SENSe{1-160}:FREQuency:STOP <numeric>
:SENSe{1-160}:FREQuency:STOP?
```

```
:SENSe{1-160}:SWEep:POINts <numeric>
:SENSe{1-160}:SWEep:POINts?
:SOURce{1-160}:POWer:PORT{1|2|3|4}[:LEVel][:IMMediate][:AMPLitude]<numeric>
:SOURce{1-160}:POWer:PORT{1|2|3|4}[:LEVel][:IMMediate][:AMPLitude]?
:SOURce{1-160}:POWer:PORT:COUPle {ON|OFF|1|0}
:SOURce{1-160}:POWer:PORT:COUPle?
:SYSTem:ERRor?
:SYSTem:PRESet
:TRIGger[:SEQuence]:SOURce {INTernal|EXTernal|MANual|BUS}
:TRIGger[:SEQuence]:SOURce?
```

Appendix B — IVI Functions

B-1 Introduction

This document is preliminary and subject to change. For further support contact Anritsu at ShockLineVNA.support@anritsu.com

Anritsu ShockLine MS46121A/B, MS46122A/B, MS46322A/B, MS46522B, and MS46524B VNAs can be remotely controlled over their Ethernet connection by means of an IVI-C driver built on a binary communications protocol rather than SCPI protocols. Anritsu ShockLine MS46121A/B and MS46122A/B VNAs can also be locally controlled through IVI-C from the externally connected PC.

This chapter documents the IVI-C functions provided by the IVI-C driver. It lists functions in alphabetical order, with a description of the function and its C syntax, a description of each parameter, and error codes where applicable. For C#, Python, MATLAB usage the usage is similar.

The general process for control is:

- connect to the instrument
- configure the instrument
- take measurements
- close the connection with the instrument when done

To do this, the following conditions must be met:

- the instrument is reachable via Ethernet
- the IP address supplied to the initialize function must match the IP address of the instrument
- ShockLine IVI Service must be running on instrument
- ANVNA IVI-C driver must be installed on client/remote PC.

Error and Status Information

Each function in the instrument driver returns a status code that indicates either success or an error condition. Your program should examine the status code from each call to an instrument driver function to determine if an error occurred.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Non-zero	Error

Command Descriptions and Syntax

Each IVI command is followed by a complete descriptive listing of the command description, parameters, and output. If applicable, an example for each command, the default value, and the range information is written out at the end of the individual description.

- See Chapter 2, “Programming the ShockLine™ Series VNA”, “Notational Conventions” on page 2-5 for definitions of parameters and notations.
- The following symbologies and parameter types are used throughout Anritsu VNA instrument programming. A subset of these are used in IVI programming on the device. Detailed descriptions of parameter types is available in “Data Transmission Methods” on page 2-9 and through the links below.
  - <NR1>
  - <NR2>

- <NR3>
- <NRf>
- <string>
- <ASCII> or <Arbitrary ASCII>
- <block> or <arbitrary block>
- <char>
- <char1>,<char2>
- <char1>,<char2>,<char3>
- <char1>,<char2>,<char3>,<char4>
- MPND
- MPNF
- MPNI

IVI Compliance Information [Agilent Source]

Compliance Category

IVI drivers come in a variety of types and configurations. This section provides IVI-required compliance information on the various categories of IVI compliance for the Agilent NA driver.

Compliance Category	Compliance Information
Category Name	IVI-COM/C Custom Specific Instrument Driver
Class Specification Version	No IVI instrument class supported

Optional Features

This section provides IVI-required information regarding optional IVI driver features supported by the AgilentNA drive.

Feature Supported

Interchangeability checking	No
State caching	No
Coercion recording	No

Driver Identification

This section provides IVI-required identity information for the AgilentNA driver.

Identification Category	**Description**
Driver Revision	1.1
Driver Vendor	Agilent Technologies
Driver DescriptionI	VI driver for Agilent Network Analyzers
Prefix/Component Identifier	AgilentNA

Supported Models **E5070B, E5071B, E5071C, E5072A, E5061A, E5062A, E5061B, E5063A, E8361A, E8362B, E8363B, E8364B, E8361C, E8362C, E8363C, E8364C, N5230A, N5230C, N5241A, N5242A, N5244A, N5245A, N5264A, N5247A, N5221A, N5222A, N5224A, N5225A, N5227A**

IVI-C Driver Information

The ANVNA IVI driver provides an IVI-C implementation. Any other programming languages must use wrapping methods to call the IVI-C driver.

Driver Identification

This section provides IVI-required identity information for the ANVNA driver.

Identification Category	Description
Driver Revision	1.0
Driver Vendor	Anritsu
Driver Description	IVI driver for Anritsu ShockLine VNAs
Prefix/Component Identifier	ANVNA
Supported Models	MS46121A/B, MS46122A/B, MS46322A/B, MS46522B, and MS46524B are supported. IVI requires V1.1.03 or later ShockLine software.

Functions and Attributes General Information

Attribute Accessors

See also: "Attributes" on page B-325.

Attribute	Attribute Accessor
Get Attribute Vi Boolean	ANVNA_GetAttributeViBoolean
Get Attribute Vi UInt 32	ANVNA_GetAttributeViUInt32
Get Attribute Vi Int32	ANVNA_GetAttributeViInt32
Get Attribute Vi Int64	ANVNA_GetAttributeViInt64
Get Attribute Vi Real64	ANVNA_GetAttributeViReal64
Get Attribute Vi String	ANVNA_GetAttributeViString
Set Attribute Vi Boolean	ANVNA_SetAttributeViBoolean
Set Attribute Vi UInt 32	ANVNA_SetAttributeViUInt32
Set Attribute Vi Int32	ANVNA_SetAttributeViInt32
Set Attribute Vi Int64	ANVNA_SetAttributeViInt64
Set Attribute Vi Real64	ANVNA_SetAttributeViReal64
Set Attribute Vi String	ANVNA_SetAttributeViString

Functions and Their Corresponding IVI Class

The following lists Anritsu IVI functions by their IVI class and Anritsu subcategory within the class.

Class	Subcategory	Function Purpose	Function Name
Channel	Measurement	Create a Measurement Using a Channel	ANVNA_ChannelMeasurementCreate
Channel	Measurement	Data To Memory	ANVNA_ChannelMeasurementDataToMemory
Channel	Measurement	Fetch Complex	ANVNA_ChannelMeasurementFetchComplex
Channel	Measurement	Fetch Formatted	ANVNA_ChannelMeasurementFetchFormatted
Channel	Measurement	Fetch Memory Complex	ANVNA_ChannelMeasurementFetchMemoryComplex
Channel	Measurement	Fetch Memory Formatted	ANVNA_ChannelMeasurementFetchMemoryFormatted
Channel	Measurement	Fetch X	ANVNA_ChannelMeasurementFetchX
Channel	Measurement	Get Channel Measurement Name	ANVNA_GetChannelMeasurementName
Channel	Measurement	Get S Parameter	ANVNA_ChannelMeasurementGetSParameter
Channel	Measurement	Set S Parameter	ANVNA_ChannelMeasurementSetSParameter
Channel	Measurement	Get Response Type	ANVNA_ChannelMeasurementGetResponseType
Channel	Measurement	Set User Defined Parameter	ANVNA_ChannelMeasurementSetUserDefinedParameter
Channel	Measurement	Get User Defined Parameter	ANVNA_ChannelMeasurementGetUserDefinedParameter
Channel	Measurement	Get Mixed Mode Response Type	ANVNA_ChannelMeasurementGetMixedModeResponseType
Channel	Measurement	Set Mixed Mode Two Diff Pairs Response	ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse
Channel	Measurement	Get Mixed Mode Two Diff Pairs Response	ANVNA_ChannelMeasurementGetMixedModeTwoDiffPairsResponse
Channel	Measurement	Set Mixed Mode One Diff Pair One Sing Response	ANVNA_ChannelMeasurementSetMixedModeOneDiffPairOneSingResponse
Channel	Measurement	Get Mixed Mode One Diff Pair One Sing Response	ANVNA_ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse
Channel	Measurement	Set Mixed Mode One Diff Pair Two Sing Response	ANVNA_ChannelMeasurementSetMixedModeOneDiffPairTwoSingResponse
Channel	Measurement	Get Mixed Mode One Diff Pair Two Sing Response	ANVNA_ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse
Channel	Measurement	Set Mixed Mode One Diff Pair Response	ANVNA_ChannelMeasurementSetMixedModeOneDiffPairResponse
Channel	Measurement	Get Mixed Mode One Diff Pair Response	ANVNA_ChannelMeasurementGetMixedModeOneDiffPairResponse
Channel	Measurement	Set Max Efficiency Response	ANVNA_ChannelMeasurementSetMaxEfficiencyResponse
Channel	Measurement	Get Max Efficiency Response	ANVNA_ChannelMeasurementGetMaxEfficiencyResponse
Channel	Misc	Asynchronous Trigger Sweep	ANVNA_ChannelAsynchronousTriggerSweep
Channel	Misc	Get Channel Name	ANVNA_GetChannelName
Channel	Segment	Add Center Span	ANVNA_ChannelSegmentAddCenterSpan
Channel	Segment	Add Start Stop	ANVNA_ChannelSegmentAddStartStop
Channel	Segment	Delete All	ANVNA_ChannelSegmentDeleteAll
Channel	StimulusRange	Configure Center Span	ANVNA_ChannelStimulusRangeConfigureCenterSpan
Channel	StimulusRange	Configure Start Stop	ANVNA_ChannelStimulusRangeConfigureStartStop
Channel	Sweep	Trigger Sweep	ANVNA_ChannelTriggerSweep
System	Open/Close	Close system	ANVNA_close
System	Open/Close	Initialize system	ANVNA_init
System	Open/Close	Initialize with options	ANVNA_InitWithOptions
System	State Control	Recall State	ANVNA_SystemRecallState
System	State Control	Save State	ANVNA_SystemSaveState
System	State Control	Wait For Operation Complete	ANVNA_SystemWaitForOperationComplete
System	State Control	Reset	ANVNA_reset

Class	Subcategory	Function Purpose	Function Name
System	State Control	Reset With Defaults	ANVNA_ResetWithDefaults
System	State Control	Revision Query	ANVNA_revision_query
System	State Control	Self Test	ANVNA_self_test
System	Status	Preset	ANVNA_StatusPreset
System	Status	Error Message	ANVNA_error_message
System	Calibration	Setup calibration type, method and ports usage	ANVNA_SetupCalibration
System	Calibration	Get calibration type	ANVNA_GetCalibrationType
System	Calibration	Get calibration method	ANVNA_GetCalibrationMethod
System	Calibration	Get calibration line	ANVNA_GetCalibrationLine
System	Calibration	Store measured data during calibration	ANVNA_CalStoreData
System	Calibration	Start calibration procedure	ANVNA_StartCalibration
System	Calibration	Abort calibration procedure	ANVNA_AbortCalibration
System	Calibration	End calibration procedure	ANVNA_EndCalibration
System	Calibration	Get calibration status (on or off)	ANVNA_GetCalStatus
System	Calibration	Set calibration status on/off	ANVNA_SetCalStatus
System	Calibration Kit	Is calibration active	ANVNA_GetCalActive
System	Calibration Kit	Load calibration kit file for the specified port	ANVNA_LoadCalKit
System	Calibration Kit	Set thru properties for the specified ports	ANVNA_SetThru
System	Calibration Kit	Get thru properties for the specified ports	ANVNA_GetThru
System	Calibration Kit	Set thru reciprocal on for the specified ports	ANVNA_SetReciprocalThru
System	Calibration Kit	Is reciprocal thru on for specified ports	ANVNA_GetReciprocalThru
System	Calibration Kit	Setup Manual Calibration	ANVNA_SetupManualCalibration
System	Calibration Kit	Set Manual Cal Kit	ANVNA_SetManualCalKit
System	Calibration Kit	Get Manual Cal Kit	ANVNA_GetManualCalKit
System	Calibration Kit	Load Calibration Kit	ANVNA_LoadCalibrationKit
	Smart Cal	Add Selected Port	ANVNA_AddSelectedPort
	Smart Cal	Set AutoCal Device	ANVNA_SetAutoCalDevice
	Smart Cal	Get AutoCal Device	ANVNA_GetAutoCalDevice
	Smart Cal	Set Smart Cal Device	ANVNA_SetSmartCalDevice
	Smart Cal	Get Smart Cal Cal Device	ANVNA_GetSmartCalCalDevice
	Smart Cal	Set Additional True	ANVNA_SetAdditionalTrue
	Smart Cal	Add One Port Connection	ANVNA_AddOnePortConnection
	Smart Cal	Add Two Port Connection	ANVNA_AddTwoPortConnection
	AutoCal	Begin Auto Cal Calibration	ANVNA_BeginAutoCalCalibration
	AutoCal	Resume Auto Cal Calibration	ANVNA_ResumeAutoCalCalibration
	AutoCal	End Auto Cal Calibration	ANVNA_EndAutoCalCalibration
Marker		Get number of instrument's markers	ANVNA_GetMarkersCount
Marker		Set marker's frequency	ANVNA_SetMarkerFrequency
Marker		Get marker's frequency	ANVNA_GetMarkerFrequency
Marker		Set type of search for specified marker	ANVNA_SetMarkerSearch
Marker		Get type of search for specified marker	ANVNA_GetMarkerSearch
Marker		Get marker's value	ANVNA_GetMarkerValue
Marker		Turn marker on or off	ANVNA_SetMarkerState
Marker		Get marker's state	ANVNA_GetMarkerState
Marker		Get Marker Up Low Value	ANVNA_GetMarkerUpLowValue
Time		Enables time domain with password	ANVNA_EnableTimeDomainOption
Time		Checks for time domain option	ANVNA_IsTimeDomainInstalled
Time		Set time domain type	ANVNA_SetTimeDomainType
Time		Get time domain type	ANVNA_GetTimeDomainType
Time		Set time domain response	ANVNA_SetTimeDomainResponse

Class	Subcategory	Function Purpose	Function Name
Time		Get time domain response	ANVNA_GetTimeDomainResponse
Time		Set time domain trip	ANVNA_SetTimeDomainTrip
Time		Get time domain trip	ANVNA_GetTimeDomainTrip
Time		Set time domain unit	ANVNA_SetTimeDomainUnit
Time		Get time domain unit	ANVNA_GetTimeDomainUnit
Time		Set time domain range using start and stop values	ANVNA_SetTimeDomainRangeStartStop
Time		Get time domain range using start and stop values	ANVNA_GetTimeDomainRangeStartStop
Time		Set time domain range center and span values	ANVNA_SetTimeDomainRangeCenterSpan
Time		Get time domain range center and span values	ANVNA_GetTimeDomainRangeCenterSpan
Time		Set time domain DC Term values	ANVNA_SetTimeDomainRangeDCTerm
Time		Get time domain DC Term values	ANVNA_GetTimeDomainRangeDCTerm
Time		Set time domain range properties	ANVNA_SetTimeDomainRangeProperties
Time		Get time domain range properties	ANVNA_GetTimeDomainRangeProperties
Time		Get Time Domain Distance Values	ANVNA_GetTimeDomainDistanceValues
Time		Get Time Domain Gate Values	ANVNA_GetTimeDomainGateValues
Time		Set time domain gate start and stop interval	ANVNA_SetTimeDomainGateStartStop
Time		Get time domain gate start and stop interval	ANVNA_GetTimeDomainGateStartStop
Time		Set time domain gate center and span interval	ANVNA_SetTimeDomainGateCenterSpan
Time		Get time domain gate center and span interval	ANVNA_GetTimeDomainGateCenterSpan
Time		Set time domain gate properties	ANVNA_SetTimeDomainGateProperties
Time		Get time domain gate properties	ANVNA_GetTimeDomainGateProperties
Limits		Set Limit Testing On Off	ANVNA_SetLimitTestingOnOff
Limits		Get Limit Testing On Off	ANVNA_GetLimitTestingOnOff
Limits		Set Limit Test Result Sign	ANVNA_SetLimitTestResultSign
Limits		Get Limit Test Result Sign	ANVNA_GetLimitTestResultSign
Limits		Clear All Limits	ANVNA_ClearAllLimits
Limits		Get Limits Count	ANVNA_GetLimitsCount
Limits		Add Limit	ANVNA_AddLimit
Limits		Set Limi	ANVNA_SetLimi
Limits		Get Limit	ANVNA_GetLimit
Limits		Set Limit Type	ANVNA_SetLimitType
Limits		Get Limit Type	ANVNA_GetLimitType
Limits		Delete Limit Segment At	ANVNA_DeleteLimitSegmentAt
Limits		Is Limit Test Pass	ANVNA_IsLimitTestPass
Limits		Get Lower Limit Buffer	ANVNA_GetLowerLimitBuffer
Limits		Get Upper Limit Buffer	ANVNA_GetUpperLimitBuffer
Limits		Get Lower Limit Fail Points Buffer	ANVNA_GetLowerLimitFailPointsBuffer
Limits		Get Upper Limit Fail Points Buffer	ANVNA_GetUpperLimitFailPointsBuffer
Limits		Get Lower Trace Lower Limit Buffer	ANVNA_GetLowerTraceLowerLimitBuffer
Limits		Get Lower Trace Upper Limit Buffer	ANVNA_GetLowerTraceUpperLimitBuffer
Limits		Get Lower Trace Lower Limit Fail Points Buffer	ANVNA_GetLowerTraceLowerLimitFailPointsBuffer
Limits		Get Lower Trace Upper Limit Fail Points Buffer	ANVNA_GetLowerTraceUpperLimitFailPointsBuffer
Ripple Limits		Set Ripple Limit Testing On Off	ANVNA_SetRippleLimitTestingOnOff
Ripple Limits		Set Ripple Limit Testing On Off	ANVNA_GetRippleLimitTestingOnOff
Ripple Limits		Set Ripple Limit Test Result Sign	ANVNA_SetRippleLimitTestResultSign
Ripple Limits		Get Ripple Limit Test Result Sign	ANVNA_GetRippleLimitTestResultSign
Ripple Limits		Add Default Ripple Limit Segment	ANVNA_AddDefaultRippleLimitSegment

Class	Subcategory	Function Purpose	Function Name
Ripple Limits		Get Ripple Limits Count	ANVNA_GetRippleLimitsCount
Ripple Limits		Delete Ripple Limit Segment	ANVNA_DeleteRippleLimitSegment
Ripple Limits		Clear All Ripple Limits	ANVNA_ClearAllRippleLimits
Ripple Limits		Add Ripple Limit Segment	ANVNA_AddRippleLimitSegment
Ripple Limits		Set Ripple Limit Values	ANVNA_SetRippleLimitValues
Ripple Limits		Get Ripple Limit Values	ANVNA_GetRippleLimitValues
Ripple Limits		Delete Ripple Limit Segment At	ANVNA_DeleteRippleLimitSegmentAt
Ripple Limits		Is Ripple Limit Test Pass	ANVNA_IsRippleLimitTestPass
Ripple Limits		Set Ripple Limit X1 Val	ANVNA_SetRippleLimitX1Val
Ripple Limits		Get Ripple Limit X1 Val	ANVNA_GetRippleLimitX1Val
Ripple Limits		Set Ripple Limit X2 Val	ANVNA_SetRippleLimitX2Val
Ripple Limits		Get Ripple Limit X2 Val	ANVNA_GetRippleLimitX2Val
Ripple Limits		Set Ripple Limit Ripple Val	ANVNA_SetRippleLimitRippleVal
Ripple Limits		Get Ripple Limit Ripple Val	ANVNA_GetRippleLimitRippleVal
Ripple Limits		Get Ripple Limit Upper Lower Values	ANVNA_GetRippleLimitUpperLowerValues
Ripple Limits		Get Ripple Limit Fail Points Buffer	ANVNA_GetRippleLimitFailPointsBuffer
Ripple Limits		Set Ripple Limit Line Active	ANVNA_SetRippleLimitLineActive
Ripple Limits		Get Ripple Limit Line Active	ANVNA_GetRippleLimitLineActive
Ripple Limits		Set Ripple Limit Lines On Off	ANVNA_SetRippleLimitLinesOnOff
Ripple Limits		Get Ripple Limit Lines On Off	ANVNA_GetRippleLimitLinesOnOff
Ripple Limits		Set Ripple Limit Ripple Value Format	ANVNA_SetRippleLimitRippleValueFormat
Ripple Limits		Get Ripple Limit Ripple Value Format	ANVNA_GetRippleLimitRippleValueFormat
Ripple Limits		Get Ripple Limit Ripple Measurement Value	ANVNA_GetRippleLimitRippleMeasurementValue

Relevant Files

Function declaration header file: ANVNA.h

Function implementation binary file: ANVNA.dll

The following sections cover functions in alphabetical order.

B-2 ANVNA_ChannelAsynchronousTriggerSweep

Description: The function initiates all the sweeps on the channel and then returns without waiting for sweep finish. User is advised to call function ANVNA_SystemWaitForOperationComplete to check if/when the sweep operation is completed.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. Pass VI_NULL or empty string if operation does not apply to a repeated capability.

Syntax Example:
```
ViStatus ANVNA_ChannelAsynchronousTriggerSweep (
vi, repCapIdentifier);
```

C++ Example:

```
      /*
       * Initialization code ...
       */
      ViStatus status = VI_SUCCESS;
      char ErrorMessage[MAX_STRING_LENGTH];
      if((status = ANVNA_ChannelAsynchronousTriggerSweep(sessionId, ":")) != VI_SUCCESS)
      {
            ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
            printf("Error %d while triggering sweep asynchronously: %s\n", status,
ErrorMessage);
            ANVNA_close(sessionId);
            exit(1);
      }

      //wait for operation to complete:
      while((status = ANVNA_SystemWaitForOperationComplete(sessionId, 10)) > VI_SUCCESS)
      {
            ;//do nothing, function call is blocking during timeout
      }
      if(status != VI_SUCCESS)
      {
            ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
            printf("Error %d while waiting for operation to complete: %s\n", status,
ErrorMessage);
            ANVNA_close(sessionId);
            exit(1);
      }
    //Data is now ready to be fetched
```

C# example:

```
            /*
             * Initialization code and sweep trigger
             */
            int status = 0;
            if ((status = ANVNA.ChannelAsynchronousTriggerSweep(sessionId, ":")) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while triggering sweep
asynchronously: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            //wait for operation to complete:
            while ((status = ANVNA.SystemWaitForOperationComplete(sessionId, 10)) > 0)
            {
                ;//do nothing, function call is blocking during timeout
            }

            if (status != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while waiting for operation to
complete: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            //Data is now ready to be fetched
```

Python example:

```
   #Initialization code ...

    status = ANVNA_ChannelAsynchronousTriggerSweep(sessionId, ":")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
        print('Error {0} while triggering sweep asynchronously: {1}\n'.format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    status = 1 #VI_FALSE
    while (status == 1):
        status = ANVNA_SystemWaitForOperationComplete(sessionId, 1000)

    if status != 0:
       ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
       print(" Error {0} waiting for operation to complete:{1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

                #Data is now ready to be fetched
```

MATLAB Example:

```
%    * Initialization code ...
        status = anvna.ChannelAsynchronousTriggerSweep(':');
        if( status ~= 0)


            [ stat, ErrorMessage ] = anvna.error_message(status);
           fprintf('Error %d while triggering sweep asynchronously: %s\n', stat, ErrorMessage);
            status = anvna.close();
            return;
        end

        %wait for operation to complete:
        status = anvna.SystemWaitForOperationComplete(10);
        while (status > 0)

            %do nothing, function call is blocking during timeout
        end
        if(status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while waiting for operation to complete: %s\n', stat,
ErrorMessage);
            status = anvna.close();
            return;

        end
%Data is now ready to be fetched
```

B-3 ANVNA_ChannelMeasurementCreate

```
ViStatus ANVNA_ChannelMeasurementCreate (
     ViSession vi,
     ViConstString repCapIdentifier,
     ViInt32 receiverPortVal,
     ViInt32 sourcePortVal);
```

Description: Creates a new measurement with specified Source and Receiver port values.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
receiverPortVal	ViInt32	Receiver port value.
sourcePortVal	ViInt32	Source port value.

C Example

```
/*
 * Initialization code ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
//Create S21 measurement on channel 1 and trace 1:
if ((status = ANVNA_ChannelMeasurementCreate(session, "CH1:Trace1", 2, 1)) != VI_SUCCESS)
{
ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while setting S21 measurement on channel 1, trace 1: %s\n", status,
ErrorMessage);
ANVNA_close(session);
exit(1);
}
```

B-4 ANVNA_ChannelMeasurementDataToMemory

```
ViStatus ANVNA_ChannelMeasurementDataToMemory (
  ViSession vi,
  ViConstString repCapIdentifier);
```

Description: Saves measurement trace data in memory.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.

C++ example:

```
   /*
    * Initialisation code and sweep trigger ...
    */
    ViStatus status = VI_SUCCESS;
    char ErrorMessage[MAX_STRING_LENGTH];
if((status = ANVNA_ChannelMeasurementDataToMemory(sessionId, "CH1:Trace1")) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while saving data to memory: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
            /*
             * Initialisation code and sweep trigger
             */
            int status = 0;
            if ((status = ANVNA.ChannelMeasurementDataToMemory(sessionId, "CH1:Trace1")) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while saving data to memory: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
```

```
     }
```

Python example:

```
    #Initialisation code and sweep trigger
    status = ANVNA_ChannelMeasurementDataToMemory(sessionId, "CH1:Trace1")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while saving data to memory: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%    * Initialisation code and sweep trigger...
        status = anvna.ChannelMeasurementDataToMemory('CH1:Trace1');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while saving data to memory: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end
```

B-5 ANVNA_ChannelMeasurementFetchFormatted

```
ViStatus ANVNA_ChannelMeasurementFetchFormatted (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 retValBufferSize,
  ViReal32 retVal[],
  ViInt32* retValActualSize);
```

Description: Returns measurement data in the current format as set by the ANVNA_ATTR_CHANNEL_MEASUREMENT_FORMAT property. Smith and Polar formats are not supported.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle obtained from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
retValBufferSize	ViInt32	Number of elements in RetVal.
retVal	ViReal32[]	Output buffer containing measurement values.
retValActualSize	ViInt32* (passed by pointer)	Actual number of elements in RetVal.

C++ example:

```
      /*
 * Initialisation code and sweep trigger ...
 * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points" ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViPReal32 measData = new ViReal32[points];
ViInt32 actualSize = 0;
// fetch measurement data from trace 1:
if ((status = ANVNA_ChannelMeasurementFetchFormatted(sessionId, "CH1:Trace1", points, measData,
&actualSize)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while fetching data on channel 1, trace 1: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Use fetched data here …

delete[] measData;
```

C# example:

```
            /*
             * Initialisation code ...
             * Trigger sweep ...
             * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points" ...
             */
            int status = 0;
            float[] measData = new float[points];
            int actualSize = 0;
            // fetch measurement data from trace 1:
           if ((status = ANVNA.ChannelMeasurementFetchFormatted(sessionId, "CH1:Trace1", (int)
points, measData, out actualSize)) != 0)

            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
              System.Console.WriteLine("Error " + status + " while fetching data on channel 1,
trace 1: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            // Use fetched data here …
```

Python example:

```
  # Initialisation code
  # Sweep trigger ...
  # Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points"

    status, RESULT, RESULT_SIZE = ANVNA_ChannelMeasurementFetchFormatted(sessionId,
"CH1:Measurement1", POINTS)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
       print('Error {0} while fetching data on channel 1, trace 1: {1}\n'.format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%    Initialisation code ...
%    Sweep trigger ...
%    Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points"

        [status, measData, actualSize] = anvna.ChannelMeasurementFetchFormatted('CH1:Trace1',
points);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while fetching data on channel 1, trace 1: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;

        end
%    Use fetched data here …
```

B-6 ANVNA_ChannelMeasurementFetchComplex

```
ViStatus ANVNA_ChannelMeasurementFetchComplex (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 pRealResponseValBufferSize,
  ViReal64 pRealResponseVal[],
  ViInt32 *pRealResponseValActualSize,
  ViInt32 pImagResponseValBufferSize,
  ViReal64 pImagResponseVal[],
  ViInt32 *pImagResponseValActualSize);
```

Description: Returns real and imaginary values of data from previously stored measurement data.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
pRealResponseValBufferSize	ViInt32	Number of elements in pRealResponseVal
pRealResponseVal	ViReal64[]	Output buffer containing real values.
pRealResponseValActualSize	ViInt32* (passed by pointer)	Actual number of elements in pRealResponseVal.
pImagResponseValBufferSize	ViInt32	Number of elements in pImagResponseVal
pImagResponseVal	ViReal64[]	Output buffer containing imaginary values.
pImagResponseValActualSize	ViInt32* (passed by pointer)	Actual number of elements in pImagResponseVal.

C++ example:

```
        /*
    * Initialisation code ...
          * Trigger sweep ...
           * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into " noPoints " ...
    */
    ViStatus status = VI_SUCCESS;
    char ErrorMessage[MAX_STRING_LENGTH];
    ViPReal64 dataReal = new ViReal64[noPoints];
    ViInt32 realActualSize = 0;
    ViPReal64 dataImag = new ViReal64[noPoints];
    ViInt32 imagActualSize = 0;

if(( status = ANVNA_ChannelMeasurementFetchComplex(sessionId, "CH1:Trace1", noPoints, dataReal,
&realActualSize, noPoints, dataImag, &imagActualSize)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while fetching complex data: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Use fetched data here …

delete[] dataReal;
delete[] dataImag;
```

C# example:

```
            /*
             * Initialisation code ...
             * Trigger sweep ...
             * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points" ...
             */
            int status = 0;
            double[] dataReal = new double[points];
            int realActualSize = 0;
            double[] dataImag = new double[points];
            int imagActualSize = 0;
            if ((status = ANVNA.ChannelMeasurementFetchComplex(sessionId, "CH1:Trace1",
(int)points, dataReal, out realActualSize, (int)points, dataImag, out imagActualSize)) != 0)

            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while fetching complex data: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            // Use fetched data here …
```

Python example:

```
    # Initialisation code ...
    # Trigger sweep ...
    # Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points" ...
    status, RESULT_REAL, RESULT_SIZE_REAL, RESULT_IMAG, RESULT_SIZE_IMAG =
ANVNA_ChannelMeasurementFetchComplex(sessionId, "CH1:Measurement1", points, points)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
        print('Error {0} while fetching complex data: {1}\n'.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%    Initialisation code ...
%    Trigger sweep ...
%    Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "noPoints"

        [status, dataReal, realActualSize,dataImag,imagActualSize] =
anvna.ChannelMeasurementFetchComplex('CH1:Trace1', noPoints, noPoints);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while fetching complex data: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end
%   Use fetched data here ...
```

B-7 ANVNA_ChannelMeasurementFetchX

```
ViStatus ANVNA_ChannelMeasurementFetchX (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 retValBufferSize,
  ViReal64 retVal[],
  ViInt32 *retValActualSize);
```

Description: Returns the stimulus values for the specified channel (list of swept frequencies).

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. Pass VI_NULL or empty string if operation does not apply to a repeated capability.
retValBufferSize	ViInt32	Number of elements in RetVal.
retVal	ViReal64[]	Output buffer containing frequency values.
retValActualSize	ViInt32* (passed by pointer)	Actual number of elements in RetVal.

C++ Example:

```
/*
 * Initialisation code ...
 * Trigger sweep ...
 * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into variable "points" ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViPReal64 freq = new ViReal64[points];
ViInt32 actualSize = 0;
// fetch the frequencies list:
if ((status = ANVNA_ChannelMeasurementFetchX(sessionId, "CH1:", points, freq, &actualSize)) !=
VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while fetching frequencies list on channel 1: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// Use frequencies list here …

delete[] freq;
```

C# example:

```
             /*
         * Initialisation code ...
         * Trigger sweep ...
         * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into variable "points" ...
         */
            int status = 0;
            double[] freq = new double[points];
        int actualSize = 0;
        // fetch the frequencies list:
        if ((status = ANVNA.ChannelMeasurementFetchX(sessionId, "CH1:",(int) points, freq, out
actualSize)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while fetching frequencies list
on channel 1: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            // Use frequencies list here …
```

Python Example:

```
    #Initialisation code ...
    #Trigger sweep ...
    #Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into variable "POINTS" ...


   status, RESULT, RESULT_SIZE = ANVNA_ChannelMeasurementFetchX(sessionId, "CH1:Measurement1",
POINTS)
    if status != 0:
        status, ErrorMessage = ANVNA_error_message(sessionId, MAX_STRING_LENGTH)
        print("Error {0} while fetching frequencies list on channel 1: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Use frequencies list here…
```

MATLAB example:

```
%   Initialisation code ...
%   Trigger sweep ...
%   Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points" ...

%       fetch the frequencies list:
        [status, freq, actualSize] = anvna.ChannelMeasurementFetchX('CH1:', points);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while fetching frequencies list on channel 1: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;

        end

%   Use frequencies list here …
```

B-8 ANVNA_ChannelMeasurementGetSParameter

```
ViStatus ANVNA_ChannelMeasurementGetSParameter (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 *pReceiverPortVal,
  ViInt32 *pSourcePortVal);
```

Description: Returns the values for Source port and Receiver port number specified in measurement.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
pReceiverPortVal	ViInt32* (passed by pointer)	Receiver port.
pSourcePortVal	ViInt32* (passed by pointer)	Source port.

C++ example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViInt32 sourcePort = 0;
ViInt32 receivPort = 0;

if ((status = ANVNA_ChannelMeasurementGetSParameter(sessionId, "CH1:Measurement1", &receivPort,
&sourcePort)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while reading source and destination port numbers: %s\n", status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Source and destination port numbers are now available in sourcePort and receivPort ...
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */

int status = 0;
int sourcePort = 0;
            int receivPort = 0;
            if ((status = ANVNA.ChannelMeasurementGetSParameter(sessionId, "CH1:Measurement1",
out receivPort, out sourcePort)) != 0)

            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while reading source and
destination port numbers: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

          // Source and destination port numbers are now available in sourcePort and receivPort
...
            System.Console.WriteLine("Source port is " + sourcePort + " && Destination port is
" + receivPort);
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status, source, receiver = ANVNA_ChannelMeasurementGetSParameter(sessionId,
"CH1:Measurement1")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while reading source and destination port numbers:
{0}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Source and destination port numbers are now available in source and receiver...
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, receivPort, sourcePort] =
anvna.ChannelMeasurementGetSParameter('CH1:Measurement1');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while reading source and destination port numbers: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;

        end

%   Source and destination port numbers are now available in sourcePort and receivePort ...
```

B-9 ANVNA_ChannelMeasurementSetSParameter

```
ViStatus ANVNA_ChannelMeasurementSetSParameter (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 newReceiverPortVal,
  ViInt32 newSourcePortVal);
```

Description: Sets the measurement parameter to specified value.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
newReceiverPortVal	ViInt32	Receiver port value.
newSourcePortVal	ViInt32	Source port value.

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

// Set S21 on CH1:Measurement1
if ((status = ANVNA_ChannelMeasurementSetSParameter(sessionId, "CH1:Measurement1", 2, 1)) !=
VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while setting source and destination port numbers: %s\n", status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Now CH1:Measurement1 is set as S21 ...
```

C# example:

```
     /*
 * Initialisation code ...
 * Measurement setup ...
 */

            int status = 0;
 // Set S21 on CH1:Measurement1
            if ((status = ANVNA.ChannelMeasurementSetSParameter(sessionId, "CH1:Measurement1",
2, 1)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + "while setting source and
destination port numbers: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            // Now CH1:Measurement1 is set as S21 ...
```

Python example:

```
    # Initialisation code ...
    # Measurement setup ...

    status = ANVNA_ChannelMeasurementSetSParameter(sessionId, "CH1:Measurement1", 2, 1)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while setting source and destination port numbers:
{1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
        print(sessionId, 'ANVNA_ChannelMeasurementSetSParameter on CH1:Measurement1', status)

    # Now CH1:Measurement1 is set as S21...
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status = anvna.ChannelMeasurementSetSParameter('CH1:Measurement1', 2, 1);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting source and destination port numbers: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;

        end

%   Now CH1:Measurement1 is set as S21 ...
ANVNA_GetCalibrationMethod
```

B-10 ANVNA_ChannelMeasurementSetUserDefinedParameter

ViStatus ANVNA_ChannelMeasurementSetUserDefinedParameter(
ViSession vi,
ViConstString repCapIdentifier,
ViUInt32 numerator,
ViUInt32 denominator,
ViUInt32 DriverPort);
:

Purpose: Set user defined measurement.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
numerator	ViUInt32	Numerator definition.
denominator	ViUInt32	Denominator definition.
DriverPort	ViUInt32	Driver port value.

Numerator and denominator possible values:

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_A1

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_A2

```
ANVNA_VAL_ANRITSU_VNA_MESUREMENT_B1
ANVNA_VAL_ANRITSU_VNA_MESUREMENT_B2
ANVNA_VAL_ANRITSU_VNA_MESUREMENT_ONE
```

C++ example:

```
ViStatus status = VI_SUCCESS;
status = ANVNA_ChannelMeasurementSetUserDefinedParameter(sessionId, “CH1:TR11”,
ANVNA_VAL_ANRITSU_VNA_MESUREMENT_A1, ANVNA_VAL_ANRITSU_VNA_MESUREMENT_B1, 1);
```

C# example:

```
int status = 0;
status = ANVNA.ChannelMeasurementSetUserDefinedParameter(sessionId, “CH1:TR1”,
ANVNA.VAL_ANRITSU_VNA_MESUREMENT_A1, ANVNA.VAL_ANRITSU_VNA_MESUREMENT_B1, 1);
```

Python example:

```
status = ANVNA_ChannelMeasurementSetUserDefinedParameter(sessionId, “CH1:TR1”,
ANVNA_VAL_ANRITSU_VNA_MESUREMENT_A1, ANVNA_VAL_ANRITSU_VNA_MESUREMENT_B1, 1)
```

MATLAB example:

status = anvna.ChannelMeasurementSetUserDefinedParameter('CH1:TR1',
anvna.VAL_ANRITSU_VNA_MESUREMENT_A1, anvna.VAL_ANRITSU_VNA_MESUREMENT_B1, 1);

B-11 ViStatus ANVNA_ChannelMeasurementGetUserDefinedParameter

ViStatus ANVNA_ChannelMeasurementSetUserDefinedParameter(
ViSession vi,
ViConstString repCapIdentifier,
ViPUInt32 numerator,
ViPUInt32 denominator,
ViPUInt32 DriverPort);
:

Purpose: Get user defined measurement.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
numerator	ViPUInt32	Numerator definition.
denominator	ViPUInt32	Denominator definition.
DriverPort	ViPUInt32	Driver port value.

Numerator and denominator possible values:

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_A1

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_A2

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_B1

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_B2

ANVNA_VAL_ANRITSU_VNA_MESUREMENT_ONE

C++ example:

```
ViStatus status = VI_SUCCESS;
ViUInt32 numerator = 0;
ViUInt32 denominator = 0;
ViUInt32 DriverPort = 0;
status = ANVNA_ChannelMeasurementGetUserDefinedParameter(sessionId, “CH1:TR11”,
&numerator, &denominator, &DriverPort);
```

C# example:

```
int status = 0;
uint numerator = 0;
uint denominator = 0;
uint DriverPort = 0;
status = ANVNA.ChannelMeasurementGetUserDefinedParameter(sessionId, “CH1:TR1”, out
numerator, out denominator, out DriverPort);
```

Python example:

```
status, numerator, denominator, DriverPort =
ANVNA_ChannelMeasurementGetUserDefinedParameter(sessionId, “CH1:TR1”)
```

MATLAB example:

```
status, numerator, denominator, DriverPort =
anvna.ChannelMeasurementGetUserDefinedParameter('CH1:TR1');
```

B-12 ANVNA_ChannelMeasurementGetResponseType

```
ViStatus ANVNA_ChannelMeasurementGetResponseType (
 ViSession vi,
 ViConstString repCapIdentifier,
 ViPUInt32 ResponseType);
```

:

Purpose: Get the response type.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
ResponseType	ViPUInt32	Response type value.

C++ example:

```
ViStatus status = VI_SUCCESS;
ViUInt32 responseType = 0;
status = ANVNA_ChannelMeasurementGetResponseType(sessionId, “CH1:TR11”,
&responseType);
```

C# example:

```
int status = 0;
uint responseType = 0;
status = ANVNA.ChannelMeasurementGetResponseType(sessionId, “CH1:TR1”, out
responseType);
```

Python example:

```
status, responseType = ANVNA_ChannelMeasurementGetResponseType(sessionId,
“CH1:TR1”)
```

MATLAB example:

```
status, responseType = anvna.ChannelMeasurementGetResponseType('CH1:TR1');
```

B-13 ANVNA_ChannelMeasurementGetMixedModeResponseType

```
ViStatus ANVNA_ChannelMeasurementGetMixedModeResponseType (
ViSession vi,
ViConstString repCapIdentifier,
ViPUInt32 ResponseType);
```

:

Purpose: Get the response type.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
ResponseType	ViPUInt32	Response type value..

C++ example:

```
ViStatus status = VI_SUCCESS;
ViUInt32 responseType = 0;
status = ANVNA_ChannelMeasurementGetMixedModeResponseType(sessionId, “CH1:TR11”,
&responseType);
```

C# example:

```
int status = 0;
uint responseType = 0;
status = ANVNA.ChannelMeasurementGetMixedModeResponseType(sessionId, “CH1:TR1”, out
responseType);
```

Python example:

```
status, responseType = ANVNA_ChannelMeasurementGetMixedModeResponseType(sessionId,
“CH1:TR1”)
```

MATLAB example:

```
status, responseType = anvna.ChannelMeasurementGetMixedModeResponseType('CH1:TR1');
```

B-14 ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse

```
ViStatus ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse (
  ViSession vi,
  ViUint32 Pair1Port1,
  ViUint32 Pair1Port2,
  ViUint32 Pair2Port1,
  ViUint32 Pair2Port2,
  ViUInt32 Response);
```

:

Purpose: Sets the measurement parameter to specified value.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
Pair1Port1	ViUInt32	Pair1 port1 value.
Pair1Port2	ViUInt32	Pair1 port2 value.
Pair2Port1	ViUInt32	Pair2 port1 value.
Pair2Port\2	ViUInt32	Pair2 port2 value.
Response	ViUInt32	Response value.

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

if ((status = ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse (sessionId,
“CH1:Measurement1”, 2, 1, 2, 1, 1)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while setting mixed mode two diff pairs respons: %s\n”, status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
```

```
}
```

C# example:

```
 /*
 * Initialisation code ...
 * Measurement setup ...
 */


            int status = 0;
            if ((status =
ANVNA.ChannelMeasurementSetMixedModeTwoDiffPairsResponse(sessionId,
“CH1:Measurement1”, 2, 1, 2, 1, 1)) != 0)
            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “while setting mixed
mode two diff pairs responses: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    # Initialisation code ...
    # Measurement setup ...


    status = ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse(sessionId,
“CH1:Measurement1”, 2, 1, 2, 1, 1)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while setting mixed mode two diff pairs response:
{1}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
        print(sessionId, 'ANVNA_ChannelMeasurementSetMixedModeTwoDiffPairsResponse
on CH1:Measurement1', status)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status =
anvna.ChannelMeasurementSetMixedModeTwoDiffPairsResponse('CH1:Measurement1', 2, 1,
2, 1, 1);
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting mixed mode two diff pairs response:
%s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end
```

B-15 ANVNA_ChannelMeasurementGetMixedModeTwoDiffPairsResponse

ViStatus ANVNA_ChannelMeasurementGetMixedModeTwoDiffPairsResponse (
ViSession vi,
ViConstString repCapIdentifier,
ViUInt32 *pPair1Port1,
ViUInt32 *pPair1Port2,
ViUInt32 *pPair2Port1,
ViUInt32 *pPair2Port2,
ViUInt32 *pResponse);

:

Purpose: Returns the values for mixed mode two diff pair response.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
Pair1Port1	ViUInt32*(passed by pointer)	Pair1 port1.
Pair1Port2	ViUInt32*	Pair1 port2.
Pair2Port1	ViUInt32*	Pair2 port1

Name	Variable Type	Description
Pair2Port\2	ViUInt32*	Pair2 port2.
Response	ViUInt32* (passed by pointer)	Response.

C++ example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViUInt32 Pair1Port1 = 0;
ViUInt32 Pair1Port2 = 0;
ViUInt32 Pair2Port1 = 0;
ViUInt32 Pair2Port2 = 0;
ViUInt32 Response = 0;

if ((status = ANVNA_ChannelMeasurementGetMixedModeTwoDiffPairsResponse (sessionId,
“CH1:Measurement1”, &Pair1Port1, &Pair1Port2, &Pair2Port1, &Pair2Port2, &Response))
!= VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while reading mixed mode two diff pairs response: %s\n”, status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */

int status = 0;
int Pair1Port1 = 0;
int Pair1Port2 = 0;
```

```
int Pair2Port1 = 0;
int Pair2Port2 = 0;
            int Response = 0;
            if ((status = ANVNA.ChannelMeasurementGetMixedModeTwoDiffPairsResponse
(sessionId, “CH1:Measurement1”, out Pair1Port1, out Pair1Port2, out Pair2Port1, out
Pair2Port2, out Response)) != 0)

            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “ while reading mixed
mode two diff pairs response: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status, pair1Port1, pair1Port2, pair2Port1, pair2Port2, response =
ANVNA_ChannelMeasurementGetMixedModeTwoDiffPairsResponse (sessionId,
“CH1:Measurement1”)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while reading mixed mode two diff pairs response:
{0}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, pair1Port1, pair1Port2, pair2Port1, pair2Port2, response] =
anvna.ChannelMeasurementGetMixedModeTwoDiffPairsResponse ('CH1:Measurement1');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
```

```
            fprintf('Error %d while reading mixed mode two diff pairs response:
%s\n', stat, ErrorMessage);
            status = anvna.close();
return;


        end
```

B-16 ANVNA_ChannelMeasurementSetMixedModeOneDiffPairOneSingResponse

ViStatus ANVNA_ChannelMeasurementSetMixedModeOneDiffPairsOneSingResponse (
ViSession vi,
ViUint32 Pair1Port1,
ViUint32 Pair1Port2,
ViUint32 Singleton1,
ViUInt32 Response);

:

Purpose: Sets the measurement parameter to specified value.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
Pair1Port1	ViUInt32	Pair1 port1 value.
Pair1Port2	ViUInt32	Pair1 port2 value.
Singleton1	ViUInt32	Singleton1.
Response	ViUInt32	Response Value

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
```

```
if ((status = ANVNA_ChannelMeasurementSetMixedMode OneDiffPairOneSingResponse
(sessionId, “CH1:Measurement1”, 2, 1, 1 , 1)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while setting mixed mode one diff pair one singleton response:
%s\n”, status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */

            int status = 0;
            if ((status = ANVNA.ChannelMeasurementSetMixedMode
OneDiffPairOneSingResponse (sessionId, “CH1:Measurement1”, 2, 1, 1, 1)) != 0)
            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “while setting mixed
mode one diff pair one singleton responses: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    # Initialisation code ...
    # Measurement setup ...

    status = ANVNA_ChannelMeasurementSetMixedModeOneDiffPairOneSingResponse
(sessionId, “CH1:Measurement1”, 2, 1, 1, 1)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
```

```
        print("Error {0} while setting mixed mode one diff pair one sigleton
response: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
        print(sessionId, 'ANVNA_ChannelMeasurementSetMixedMode
OneDiffPairOneSingResponse on CH1:Measurement1', status)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status = anvna.ChannelMeasurementSetMixedMode OneDiffPairOneSingResponse
('CH1:Measurement1', 2, 1, 1, 1);
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting mixed mode one diff pair one singleton
response: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end
```

B-17 ANVNA_ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse

ViStatus ANVNA_ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse (
ViSession vi,
ViConstString repCapIdentifier,
ViUInt32 *pPair1Port1,
ViUInt32 *pPair1Port2,
ViUInt32 *pSingleton1,
ViUInt32 *pResponse);

:

Purpose: Returns the values for mixed mode one diff pair one singleton response.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
pPair1Port1	ViUInt32* (passed by pointer)	Pair1 port1.
pPair1Port2	ViUInt32*	Pair1 port2.
pSingleton1	ViUInt32*	Pair2 port1.
Pair2Port2	ViUInt32*	Singleton1.
Response	ViUInt32* (passed by pointer)	Response.

C++ example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViUInt32 Pair1Port1 = 0;
ViUInt32 Pair1Port2 = 0;
ViUInt32 Singleton1 = 0;
ViUInt32 Response = 0;

if ((status = ANVNA_ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse
(sessionId, “CH1:Measurement1”, &Pair1Port1, &Pair1Port2, &Singleton1, &Response))
!= VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while reading mixed mode one diff pair one singleton response:
%s\n”, status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
    /*
```

```
 * Initialisation code ...
 * Measurement setup ...
 */


int status = 0;
int Pair1Port1 = 0;
int Pair1Port2 = 0;
int Singleton1 = 0;
int Response = 0;
            if ((status =
ANVNA.ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse (sessionId,
“CH1:Measurement1”, out Pair1Port1, out Pair1Port2, out Singleton1, out Response))
!= 0)


            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “ while reading mixed
mode one diff pair one singleton response: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...


    status, pair1Port1, pair1Port2, singleton1, response =
ANVNA_ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse (sessionId,
“CH1:Measurement1”)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while reading mixed mode one diff pair one singleton
response: {0}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, pair1Port1, pair1Port2, singleton1, response] =
anvna.ChannelMeasurementGetMixedModeOneDiffPairOneSingResponse
('CH1:Measurement1');
        if( status ~= 0)


            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while reading mixed mode one diff pair one singleton
response: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;


        end
```

B-18 ANVNA_ChannelMeasurementSetMixedModeOneDiffPairTwoSingResponse

ViStatus ANVNA_ChannelMeasurementSetMixedModeOneDiffPairTwoSingResponse (
ViSession vi,
ViUint32 Pair1Port1,
ViUint32 Pair1Port2,
ViUint32 Singleton1,
VIUInt32 Singleton2,
ViUInt32 Response);

:

Purpose: Sets the measurement parameter to specified value.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
Pair1Port1	ViUInt32	Pair1 port1 value.
Pair1Port2	ViUInt32	Pair1 port2 value.
Singleton1	ViUInt32	Singleton1.

Name	Variable Type	Description
Singleton2	ViUInt32	Singleton2.
Response	ViUInt32	Response value.

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

if ((status = ANVNA_ChannelMeasurementSetMixedMode OneDiffPairTwoSingResponse
(sessionId, “CH1:Measurement1”, 2, 1, 1, 2, 2)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while setting mixed mode one diff pair two singleton response:
%s\n”, status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */

            int status = 0;
            if ((status = ANVNA.ChannelMeasurementSetMixedMode
OneDiffPairTwoSingResponse (sessionId, “CH1:Measurement1”, 2, 1, 1, 2, 2)) != 0)
            {
                string ErrorMessage = ““;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “while setting mixed
mode one diff pair two singleton responses: “ + ErrorMessage);
                ANVNA.close(sessionId);
```

```
                System.Environment.Exit(1);
            }
```

Python example:

```
    # Initialisation code ...
    # Measurement setup ...

    status = ANVNA_ChannelMeasurementSetMixedModeOneDiffPairTwoSingResponse
(sessionId, “CH1:Measurement1”, 2, 1, 1, 2, 2)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while setting mixed mode one diff pair two sigleton
response: {1}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
        print(sessionId, 'ANVNA_ChannelMeasurementSetMixedMode
OneDiffPairTwoSingResponse on CH1:Measurement1', status)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status = anvna.ChannelMeasurementSetMixedMode OneDiffPairTwoSingResponse
('CH1:Measurement1', 2, 1, 1, 2, 2);
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting mixed mode one diff pair two singleton
response: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end
```

B-19 ANVNA_ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse

```
ViStatus ANVNA_ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse (
     ViSession vi,
     ViConstString repCapIdentifier,
     ViUInt32 *pPair1Port1,
     ViUInt32 *pPair1Port2,
     ViUInt32 *pSingleton1,
     ViUInt32 *pSingleton2,
     ViUInt32 *pResponse);
```

:

Purpose: Returns the values for mixed mode one diff pair two singleton response.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
pPair1Port1	ViUInt32* (passed by pointer)	Pair1 port1.
pPair1Port2	ViUInt32*	Pair1 port2.
pSingleton1	ViUInt32*	Singleton1.
pSingleton2	ViUInt32*	Singleton2.
Response	ViUInt32* (passed by pointer)	Response.

C++ example:

```
   /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViUInt32 Pair1Port1 = 0;
ViUInt32 Pair1Port2 = 0;
ViUInt32 Singleton1 = 0;
ViUInt32 Singleton2 = 0;
```

```
ViUInt32 Response = 0;


if ((status = ANVNA_ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse
(sessionId, “CH1:Measurement1”, &Pair1Port1, &Pair1Port2, &Singleton1, &Singleton2,
&Response)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while reading mixed mode one diff pair two singleton response:
%s\n”, status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */


int status = 0;
int Pair1Port1 = 0;
int Pair1Port2 = 0;
int Singleton1 = 0;
int Singleton2 = 0;
int Response = 0;
            if ((status =
ANVNA.ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse (sessionId,
“CH1:Measurement1”, out Pair1Port1, out Pair1Port2, out Singleton1, out Singleton2,
out Response)) != 0)


            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “ while reading mixed
mode one diff pair two singleton response: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status, pair1Port1, pair1Port2, singleton1, singleton2, response =
ANVNA_ChannelMeasurementGetMixedModeOneDiffPaiTwoSingResponse (sessionId,
“CH1:Measurement1”)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while reading mixed mode one diff pair two singleton
response: {0}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, pair1Port1, pair1Port2, singleton1, singleton2, response] =
anvna.ChannelMeasurementGetMixedModeOneDiffPairTwoSingResponse
('CH1:Measurement1');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while reading mixed mode one diff pair two singleton
response: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end
```

B-20 ANVNA_ChannelMeasurementSetMixedModeOneDiffPairResponse

```
ViStatus ANVNA_ChannelMeasurementSetMixedModeOneDiffPairsResponse (
    ViSession vi,
    ViUint32 Pair1Port1,
    ViUint32 Pair1Port2,
    ViUInt32 Response);
```

:

Purpose: Sets the measurement parameter to specified value.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
Pair1Port1	ViUInt32	Pair1 port1 value.
Pair1Port2	ViUInt32	Pair1 port2 value.
Response	ViUInt32	Response value.

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

if ((status = ANVNA_ChannelMeasurementSetMixedModeOneDiffPairResponse (sessionId,
“CH1:Measurement1”, 2, 1, 1)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while setting mixed mode one diff pair response: %s\n”, status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
```

```
            int status = 0;
            if ((status = ANVNA.ChannelMeasurementSetMixedMode OneDiffPairResponse
(sessionId, “CH1:Measurement1”, 2, 1, 1)) != 0)
            {
                string ErrorMessage = ““;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “while setting mixed
mode one diff pair response: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    # Initialisation code ...
    # Measurement setup ...

    status = ANVNA_ChannelMeasurementSetMixedModeOneDiffPairResponse (sessionId,
“CH1:Measurement1”, 2, 1, 1)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while setting mixed mode one diff pair response:
{1}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
        print(sessionId, 'ANVNA_ChannelMeasurementSetMixedMode OneDiffPairResponse
on CH1:Measurement1', status)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status = anvna.ChannelMeasurementSetMixedModeOneDiffPairResponse
('CH1:Measurement1', 2, 1, 1);
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting mixed mode one diff pair response:
%s\n', stat, ErrorMessage);
            status = anvna.close();
```

```
return;

        end
```

B-21 ANVNA_ChannelMeasurementGetMixedModeOneDiffPairResponse

```
ViStatus ANVNA_ChannelMeasurementGetMixedModeOneDiffPairResponse (
    ViSession vi,
    ViConstString repCapIdentifier,
    ViUInt32 *pPair1Port1,
    ViUInt32 *pPair1Port2,
    ViUInt32 *pResponse);
```

:

Purpose: Returns the values for mixed mode one diff pair response.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
pPair1Port1	ViUInt32* (passed by pointer)	Pair1 port1.
pPair1Port2	ViUInt32*	Pair1 port2.
Response	ViUInt32* (passed by pointer)	Response.

C++ example:

```
   /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViUInt32 Pair1Port1 = 0;
ViUInt32 Pair1Port2 = 0;
ViUInt32 Response = 0;
```

```
if ((status = ANVNA_ChannelMeasurementGetMixedModeOneDiffPairResponse (sessionId,
“CH1:Measurement1”, &Pair1Port1, &Pair1Port2, &Response)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while reading mixed mode one diff pair response: %s\n”, status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */

int status = 0;
int Pair1Port1 = 0;
int Pair1Port2 = 0;
int Response = 0;
            if ((status = ANVNA.ChannelMeasurementGetMixedModeOneDiffPairResponse
(sessionId, “CH1:Measurement1”, out Pair1Port1, out Pair1Port2, out Response)) !=
0)

            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “ while reading mixed
mode one diff pair response: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...
```

```
    status, pair1Port1, pair1Port2, response =
ANVNA_ChannelMeasurementGetMixedModeOneDiffPairResponse (sessionId,
“CH1:Measurement1”)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while reading mixed mode one diff pair response:
{0}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, pair1Port1, pair1Port2, response] =
anvna.ChannelMeasurementGetMixedModeOneDiffPairResponse ('CH1:Measurement1');
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while reading mixed mode one diff pair response:
%s\n', stat, ErrorMessage);
            status = anvna.close();
return;
       end
```

B-22 ANVNA_ChannelMeasurementSetMaxEfficiencyResponse

```
ViStatus ANVNA_ChannelMeasurementSetMaxEfficiencyResponse(
    ViSession vi,
    ViUint32 Port1,
    ViUint32 Port2);
```

:

Purpose: Sets the measurement parameter to specified value.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.

Name	Variable Type	Description
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
Pair1Port1	ViUInt32	Pair1 value.
Pair1Port2	ViUInt32	Pair2 value.

C++ example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

if ((status = ANVNA_ChannelMeasurementSetMaxEfficiencyResponse (sessionId,
“CH1:Measurement1”, 1, 2)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while setting max efficiency response: %s\n”, status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */

            int status = 0;
            if ((status = ANVNA.ChannelMeasurementSetMaxEfficiencyResponse
(sessionId, “CH1:Measurement1”, 1, 2)) != 0)
            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “while setting max
efficiency response: “ + ErrorMessage);
```

```
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    # Initialisation code ...
    # Measurement setup ...

    status = ANVNA_ChannelMeasurementSetMaxEfficiencyResponse (sessionId,
“CH1:Measurement1”, 1, 2)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while setting max efficiency response:
{1}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
        print(sessionId, 'ANVNA_ChannelMeasurementSetMaxEfficiencyResponse on
CH1:Measurement1', status)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status = anvna.ChannelMeasurementSetMaxEfficiencyResponse
('CH1:Measurement1', 1, 2);
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting max efficiency response: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;

        end
```

B-23 ANVNA_ChannelMeasurementGetMaxEfficiencyResponse

```
ViStatus ANVNA_ChannelMeasurementGetMaxEfficiencyResponse (
    ViSession vi,
    ViConstString repCapIdentifier,
    ViUInt32 *pPort1,
    ViUInt32 *pPort2);
```

:

Purpose: Returns the values for max efficiency response.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
pPair1Port1	ViUInt32* (passed by pointer)	Pair1.
pPair1Port2	ViUInt32*	Pair2.

C++ example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
ViUInt32 Port1 = 0;
ViUInt32 Port2 = 0;

if ((status = ANVNA_ChannelMeasurementGetMaxEfficiencyResponse (sessionId,
“CH1:Measurement1”, &Port1, &Port2)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf(“Error %d while reading max efficiency response: %s\n”, status,
ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */

int status = 0;
int Port1 = 0;
int Port2 = 0;
if ((status = ANVNA.ChannelMeasurementGetMaxEfficiencyResponse (sessionId,
“CH1:Measurement1”, out Port1, out Port2)) != 0)
            {
                string ErrorMessage = “”;
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine(“Error “ + status + “ while reading max
efficiency response: “ + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status, port1, port2 = ANVNA_ChannelMeasurementGetMaxEfficiencyResponse
(sessionId, “CH1:Measurement1”)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(“Error {0} while reading max efficiency response:
{0}\n”.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, port1, port2] = anvna.ChannelMeasurementGetMaxEfficiencyResponse
('CH1:Measurement1');
        if( status ~= 0)
            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while reading max efficiency response: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;
       end
```

B-24 ANVNA_ChannelSegmentGetProperties

```
ViStatus ANVNA_ ChannelSegmentGetProperties (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 SegmentId,
  ViPReal64 StartVal,
  ViPReal64 StopVal,
  ViPInt32 NumberOfPointsVal,
  ViPReal64 IFBandwidthVal);
```

Purpose: Returns all properties (start and stop frequencies, number of points, IF Bandwidth, Power and time values) of a given segment. Segment is selected using segmented value.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:”)	ViConstString	The physical or virtual repeated capability identifie
SegmentId	ViInt32	Segment selector. Range is 0:ANVNA_ChannelSegmentGetCount Result -1
StartVal	ViPReal64	Source port value.
StopVal	ViPReal64	Stop frequency [Hz]
NumberOfPointsVal	ViPInt32	Number of points
IFBandwidthVal	ViPReal64	IF Bandwidth [Hz]

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

ViReal64 StartVal;
ViReal64 StopVal;
ViInt32 NumberOfPointsVal;
ViReal64 IFBandwidthVal;
    if(( status = ANVNA_ChannelSegmentGetProperties(sessionId, "CH1:", 1, &StartVal, &StopVal,
&NumberOfPointsVal, &IFBandwidthVal)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while getting segment properties: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Now there is a new segment on channel CH1
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
int status = 0;
double StartVal;
double StopVal;
int NumberOfPointsVal;
double IFBandwidthVal;

if ((status = ANVNA.ChannelSegmentGetProperties(sessionId, "CH1:", 1, out StartVal, out
StopVal, out NumberOfPointsVal, out IFBandwidthVal)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while getting segment properties:
" + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            // Now there is a new segment on channel CH1
```

Python example:

```
    #Initialisation code...
    #Measurement setup...

    Status, StartVal, StopVal, NumberOfPointsVal, IFBandwidthVal =
ANVNA_ChannelSegmentGetProperties(sessionId, "CH1: ", 1)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while getting segment properties: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Now there is a new segment on channel CH1
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        [status, StartVal, StopVal, NumberOfPointsVal, IFBandwidthVal] =
anvna.ChannelSegmentGetProperties('CH1:', 1);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while getting segment properties: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;
        end
```

B-25 ANVNA_ChannelSegmentAddCenterSpan

```
ViStatus ANVNA_ChannelSegmentAddCenterSpan (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViReal64 centerVal,
  ViReal64 spanVal,
  ViInt32 numberOfPointVal,
  ViReal64 IFBandwidthVal;
```

Description: Adds a new segment using center and span frequency values.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.
centerVal	ViReal64	Center frequency in Hz.
spanVal	ViReal64	Span value, in Hz.
numberOfPointVal	ViInt32	Number of points. Maximum number depends on VNA model; see specifications.
IFBandwidthVal	ViReal64	IF Bandwidth, in Hz. Allowed values depend on VNA model; see specifications.

B-26 ANVNA_ChannelSegmentAddStartStop

```
ViStatus ANVNA_ChannelSegmentAddStartStop (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViReal64 startVal,
  ViReal64 stopVal,
  ViInt32 numberOfPointsVal,
  ViReal64 IFBandwidthVal;
```

Description: Adds a new segment using start and stop frequency values.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.
startVal	ViReal64	Start frequency in Hz.
stopVal	ViReal64	Stop frequency in Hz.
numberOfPointVal	ViInt32	Number of points. Maximum number depends on VNA model; see specifications.
IFBandwidthVal	ViReal64	IF Bandwidth in Hz. Maximum value depends on VNA model; see specifications.

C++ example:

```
      /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

    if(( status = ANVNA_ChannelSegmentAddStartStop(sessionId, "CH1:", 1000000000.0,
7000000000.0, 401, 10000.0)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while adding a new segment: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Now there is a new segment on channel CH1
```

C# example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
           int status = 0;
if ((status = ANVNA.ChannelSegmentAddStartStop(sessionId, "CH1:", 1000000000.0, 7000000000.0,
401, 10000.0)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while adding a new segment: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            // Now there is a new segment on channel CH1
```

Python example:

```
    #Initialisation code...
    #Measurement setup...

    status = ANVNA_ChannelSegmentAddStartStop(sessionId, "CH1: ", float(1000000000),
float(7000000000), 401, 10000)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print(sessionId, 'ANVNA_ChannelSegmentAddStartStop on CH1:Measurement1', status)
        print("Error {0} while adding a new segment: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Now there is a new segment on channel CH1
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...
        status = anvna.ChannelSegmentAddStartStop('CH1:', 1000000000.0, 7000000000.0, 401,
10000.0);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while adding a new segment: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

%   Now there is a new segment on channel CH1


            status = anvna.close();
return;

        end

%   Now there is a new segment on channel CH1
```

B-27 ANVNA_ChannelSegmentDeleteAll

```
ViStatus ANVNA_ChannelSegmentDeleteAll (
  ViSession vi,
  ViConstString repCapIdentifier);
```

Description: Deletes all the segments for a channel.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.

C++ example:

```
     /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

    if(( status = ANVNA_ChannelSegmentDeleteAll(sessionId, "CH1:")) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while deleting all segments: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// Now CH1 is not segmented
```

C# example:

```
/*
 * Initialisation code ...
 * Measurement setup ...
 */
int status = 0;
           if ((status = ANVNA.ChannelSegmentDeleteAll(sessionId, "CH1:")) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while deleting all segments: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
// Now CH1 is not segmented
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status = ANVNA_ChannelSegmentDeleteAll(sessionId, "CH1:")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while deleting all segments: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Now CH1 is not segmented
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...

        status = anvna.ChannelSegmentDeleteAll('CH1:');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while deleting all segments: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

%   Now CH1 is not segmented
```

B-28 ANVNA_SetupManualCalibration

```
ViStatus ANVNA_SetupManualCalibration (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViUInt32 calibrationMethod,
  ViUInt32 calibrationLine);
```

Purpose: Setup a manual calibration process. This is the initial step in performing a calibration.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:”)	ViConstString	The physical or virtual repeated capability identifier.
calibrationMethod	ViUInt32	The calibration method, see possible values below
calibrationLine	ViUInt32	The calibration line, see possible values below

calibrationMethod values:

ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL	11

calibrationLine VALUES:

ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL	0
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_WAVEGUIDE	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_MICROSTRIP	2
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_NONDISPERSIVE	3

C/C++ example:

```
/* Start a full two port calibration for port one and two */
status = ANVNA_SetupManualCalibration(sessionId, “CH1:”,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL);
```

C# example:

```
/* Start a full two port calibration for port one and two */
status = ANVNA.SetupManualCalibration(sessionId, “CH1:”,
ANVNA.VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT,
ANVNA.VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL);
```

Python example:

```
# Start a full two port calibration for port one and two
```

```
status = ANVNA_SetupManualCalibration(sessionId, “CH1:”,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL)
```

MATLAB example:

```
% Start a full two port calibration for port one and two
status = anvna.SetupManualCalibration('CH1:',
anvna.VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT,
anvna.VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL);
```

B-29 ANVNA_SetManualCalKit

```
ViStatus ANVNA_SetManualCalKit (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViUInt16 PortNumber,
  ViUInt32 KitId,
  ViUInt32 BBLoad);
```

Purpose: Setup a manual calibration kit from the predefined list.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:”)	ViConstString	The physical or virtual repeated capability identifier.
PortNumber	ViUInt16	The port number.
KitId	ViUInt32	The kit id definition.
BBLoad	ViUInt32	BB load parameter.

KitId values

ANVNA_VAL_ANRITSU_VNA_CALKIT_TWOPOINTFOUR_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPCTHREEPOINTFIVE_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_KCONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_SMA_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TNC_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_VCONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_W1CONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_SEVENSIXTEEN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPC7_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN75_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLK50_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLN50_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GCS35M_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR10_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR12_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR15_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR28_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR42_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR62_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR75_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR90_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR112_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR137_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR159_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR187_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR229_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED1_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED2_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED3_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED4_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED5_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED6_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED7_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED8_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TWOPOINTFOUR_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPCTHREEPOINTFIVE_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_KCONN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN_FEMALE

ANVNA_VAL_ANRITSU_VNA_CALKIT_SMA_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TNC_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_VCONN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_W1CONN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_SEVENSIXTEEN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPC7_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN75_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLK50_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLN50_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GCS35M_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR10_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR12_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR15_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR28_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR42_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR62_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR75_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR90_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR112_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR137_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR159_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR187_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR229_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED1_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED2_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED3_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED4_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED5_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED6_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED7_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED8_FEMALE

C/C++ example:

```
/* Start a full two port calibration for port one and two */
status = ANVNA_SetManualCalKit(sessionId, “CH1:TR1”, 1,
ANVNA_VAL_ANRITSU_VNA_CALKIT_KCONN_MALE, 0);
```

C# example:

```
/* Start a full two port calibration for port one and two */
```

```
status = ANVNA.SetManualCalKit(sessionId, “CH1:TR1”, 1,
ANVNAVAL_ANRITSU_VNA_CALKIT_KCONN_MALE, 0);
```

Python example:

```
# Start a full two port calibration for port one and two
status = ANVNA_SetManualCalKit(sessionId, “CH1:TR1”, 1,
ANVNA_VAL_ANRITSU_VNA_CALKIT_KCONN_MALE, 0)
```

MATLAB example:

```
% Start a full two port calibration for port one and two
status = anvna.SetManualCalKit('CH1:TR1', 1,
anvna.VAL_ANRITSU_VNA_CALKIT_KCONN_MALE, 0);
```

B-30 ANVNA_ChannelSegmentGetCount

```
ViStatus ANVNA_ChannelSegmentGetCount (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt32 numSegments);
```

Description: Returns the number of segments declared per channel. Default value is 1. The function will return 1 if no segments are declared or after ANVNA_ChannelSegmentDeleteAll is called.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	(Only channel index is parsed, ex. “CH1:”) ViConstString The physical or virtual repeated capability identifier.
numSegments	ViPUInt32	Number of segments returned by the function

C++ Example

```
      // get number of points:
      ViUInt32 segments = 0;
      if ((status = ANVNA_ChannelSegmentGetCount(sessionId, “CH1:”, &segments)) != VI_SUCCESS)
      {
char ErrorMessage[MAX_STRING_LENGTH];
      ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
      printf("Error %d while retrieving number of segments: %s\n", status, ErrorMessage);
      ANVNA_close(sessionId);
      exit(1);
      }
printf(“number of segments is %d\n”, segments);
```

C# Example:

```
       // Get the number of points
            uint segments;
            if ((status = ANVNA.ChannelSegmentGetCount(sessionId, "CH1:", out segments)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while retrieving number of
segments:  " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            System.Console.WriteLine("number of segments " + segments);
```

Python Example:

```
    # Get the number of points:
    status, segments =  ANVNA_ChannelSegmentGetCount(sessionId, "CH1:Measurement1")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
        print('Error {0} while retrieving number of segments: {1}\n'.format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    print ("number of segments is {0}\n".format(segments))
```

MATLAB Example:

```
%   get number of points:
    [status, segments] = anvna.ChannelSegmentGetCount('CH1:');
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while retrieving number of segments: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end

    fprintf('number of segments is %d\n', segments);
```

B-31 ANVNA_GetManualCalKit

```
ViStatus ANVNA_GetManualCalKit (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViUInt16 PortNumber,
  ViPUInt32 KitId,
  ViPUInt32 BBLoad);
```

Purpose: Get manual calibration kit.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(Only channel index is parsed, ex. “CH1:”)	ViConstString	The physical or virtual repeated capability identifier.
PortNumber	ViUInt16	The port number.
KitId	ViPUInt32	The kit id.
BBLoad	ViPUInt32	BB load parameter.

KitId values:

ANVNA_VAL_ANRITSU_VNA_CALKIT_TWOPOINTFOUR_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPCTHREEPOINTFIVE_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_KCONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_SMA_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TNC_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_VCONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_W1CONN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_SEVENSIXTEEN_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPC7_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN75_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLK50_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLN50_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GCS35M_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR10_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR12_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR15_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR28_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR42_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR62_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR75_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR90_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR112_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR137_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR159_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR187_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR229_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED1_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED2_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED3_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED4_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED5_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED6_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED7_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED8_MALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TWOPOINTFOUR_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPCTHREEPOINTFIVE_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_KCONN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN_FEMALE

ANVNA_VAL_ANRITSU_VNA_CALKIT_SMA_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TNC_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_VCONN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_W1CONN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_SEVENSIXTEEN_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GPC7_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_NCONN75_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLK50_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_TOSLN50_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_GCS35M_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR10_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR12_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR15_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR28_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR42_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR62_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR75_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR90_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR112_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR137_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR159_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR187_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_WR229_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED1_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED2_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED3_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED4_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED5_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED6_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED7_FEMALE
ANVNA_VAL_ANRITSU_VNA_CALKIT_USERDEFINED8_FEMALE

C/C++ example:

```
ViUInt32 KitId;
ViUInt32 BBLoad;

status = ANVNA_GetManualCalKit(sessionId, “CH1:TR1”, 1, KitId, BBLoad);
```

C# example:

```
uint KitId;
uint BBLoad;

status = ANVNA.SetManualCalKit(sessionId, “CH1:TR1”, 1, out KitId, out BBLoad);
```

Python example:

```
status, KitId, BBLoad = ANVNA_SetManualCalKit(sessionId, “CH1:TR1”, 1)
```

MATLAB example:

```
status, KitId, BBLoad = anvna.SetManualCalKit('CH1:TR1', 1);
```

B-32 ANVNA_LoadCalibrationKit

```
ViSession vi,
ViConstString RepCapIdentifier,
ViConstString calKitFile,
ViConstString label,
ViUInt32 type,
ViUInt32 calibrationLine,
ViUInt32 calibrationMethod);
```

Purpose: Loads a calibration kit from file.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:”)	ViConstString	(Only channel index is parsed, ex. “CH1:”)ViConstString The physical or virtual repeated capability identifier.
calKitFile	ViConstString	The CCF file name to be loaded.
label	ViConstString	The GUI label associated with this.
type	ViUInt32	Calibration kit type.
calibrationLine	ViUInt32	Calibration line.
calibrationMethod	ViUInt32	Calibration method.

type values:

ANVNA_VAL_ANRITSU_VNA_CALKIT_TYPE_CCF
ANVNA_VAL_ANRITSU_VNA_CALKIT_TYPE_S1P

calibrationLine values

ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_WAVEGUIDE
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_MICROSTRIP
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_NONDISPERSIVE:

calibrationMethod values:

ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL

Examples

C++ example:

```
ANVNA_LoadCalibrationKit(sessionId, “CH1:”, “C:\AnritsuVNA\custom.ccf”, “Custom
Kit”, ANVNA_VAL_ANRITSU_VNA_CALKIT_TYPE_CCF,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT);
```

C# example:

```
ANVNA.LoadCalibrationKit(sessionId, “CH1:”, “C:\AnritsuVNA\custom.ccf”, “Custom
Kit”, ANVNA.VAL_ANRITSU_VNA_CALKIT_TYPE_CCF,
ANVNA.VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL,
ANVNA.VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT);
```

Python example:

```
ANVNA_LoadCalibrationKit(sessionId, “CH1:”, “C:\AnritsuVNA\custom.ccf”, “Custom
Kit”, ANVNA_VAL_ANRITSU_VNA_CALKIT_TYPE_CCF,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT)
```

MATLAB example:

```
anvna.LoadCalibrationKit(sessionId, “CH1:”, “C:\AnritsuVNA\custom.ccf”, “Custom
Kit”, anvna.VAL_ANRITSU_VNA_CALKIT_TYPE_CCF,
anvna.VAL_ANRITSU_VNA_CALIBRATION_LINETYPE_COAXIAL,
anvna.VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT);
```

B-33 ANVNA_SetupCalibration

ViStatus ANVNA_SetupCalibration (

ViSession vi,

ViConstString RepCapIdentifier,

Bool usePort1,

Bool usePort2,

Bool usePort3,

Bool usePort4,

ViUInt32 calibrationType,

ViUInt32 calibrationMethod);

Description: Setup a calibration process.

Parameter List:

Name	Variable Type	Description
vi	ViSession	ViSession The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	(Only channel index is parsed, ex. “CH1:”) ViConstString The physical or virtual repeated capability identifier.
usePort1		Bool Sets port 1 either active or inactive for the current calibration setup (True = Active)
usePort2		Bool Sets port 2 either active or inactive for the current calibration setup (True = Active)
usePort3		Bool Sets port 3 either active or inactive for the current calibration setup (True = Active)
usePort4		Bool Sets port 4 either active or inactive for the current calibration setup (True = Active)
calibrationType	ViUInt32	The calibration type
calibrationMethod	ViUInt32	The calibration type

calibrationType	Values
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL	11

calibrationMethod	Values
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL	11

Example: See ANVNA_StartCalibration.

B-34 ANVNA_ChannelSegmentDeleteAll

```
ViStatus ANVNA_ChannelSegmentDeleteAll (
  ViSession vi,
  ViConstString repCapIdentifier);
```

Description: Deletes all the segments for a channel.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.

C++ example:

```
     /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

    if(( status = ANVNA_ChannelSegmentDeleteAll(sessionId, "CH1:")) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while deleting all segments: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// Now CH1 is not segmented
```

C# example:

```
/*
 * Initialisation code ...
 * Measurement setup ...
 */
int status = 0;
           if ((status = ANVNA.ChannelSegmentDeleteAll(sessionId, "CH1:")) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while deleting all segments: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
// Now CH1 is not segmented
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status = ANVNA_ChannelSegmentDeleteAll(sessionId, "CH1:")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while deleting all segments: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Now CH1 is not segmented
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...

        status = anvna.ChannelSegmentDeleteAll('CH1:');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while deleting all segments: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

%   Now CH1 is not segmented
```

B-35 ANVNA_ChannelSegmentGetCount

```
ViStatus ANVNA_ChannelSegmentGetCount (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt32 numSegments);
```

Description: Returns the number of segments declared per channel. Default value is 1. The function will return 1 if no segments are declared or after ANVNA_ChannelSegmentDeleteAll is called.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	(Only channel index is parsed, ex. “CH1:”) ViConstString The physical or virtual repeated capability identifier.
numSegments	ViPUInt32	Number of segments returned by the function

C++ Example

```
      // get number of points:
      ViUInt32 segments = 0;
      if ((status = ANVNA_ChannelSegmentGetCount(sessionId, “CH1:”, &segments)) != VI_SUCCESS)
      {
char ErrorMessage[MAX_STRING_LENGTH];
      ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
      printf("Error %d while retrieving number of segments: %s\n", status, ErrorMessage);
      ANVNA_close(sessionId);
      exit(1);
      }
printf(“number of segments is %d\n”, segments);
```

C# Example:

```
       // Get the number of points
            uint segments;
            if ((status = ANVNA.ChannelSegmentGetCount(sessionId, "CH1:", out segments)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while retrieving number of
segments:  " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            System.Console.WriteLine("number of segments " + segments);
```

Python Example:

```
    # Get the number of points:
    status, segments =  ANVNA_ChannelSegmentGetCount(sessionId, "CH1:Measurement1")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
        print('Error {0} while retrieving number of segments: {1}\n'.format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    print ("number of segments is {0}\n".format(segments))
```

MATLAB Example:

```
%   get number of points:
    [status, segments] = anvna.ChannelSegmentGetCount('CH1:');
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while retrieving number of segments: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end

    fprintf('number of segments is %d\n', segments);
```

B-36 ANVNA_SetupCalibration

ViStatus ANVNA_SetupCalibration (

ViSession vi,

ViConstString RepCapIdentifier,

Bool usePort1,

Bool usePort2,

Bool usePort3,

Bool usePort4,

ViUInt32 calibrationType,

ViUInt32 calibrationMethod);

Description: Setup a calibration process.

Parameter List:

Name	Variable Type	Description
vi	ViSession	ViSession The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	(Only channel index is parsed, ex. “CH1:”) ViConstString The physical or virtual repeated capability identifier.
usePort1		Bool Sets port 1 either active or inactive for the current calibration setup (True = Active)
usePort2		Bool Sets port 2 either active or inactive for the current calibration setup (True = Active)
usePort3		Bool Sets port 3 either active or inactive for the current calibration setup (True = Active)
usePort4		Bool Sets port 4 either active or inactive for the current calibration setup (True = Active)
calibrationType	ViUInt32	The calibration type
calibrationMethod	ViUInt32	The calibration type

calibrationType	Values
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL	11

calibrationMethod	Values
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL	11

Example: See ANVNA_StartCalibration.

B-37 ANVNA_StartCalibration

```
ViStatus ANVNA_StartCalibration
  ViSession vi,
  ViConstString RepCapIdentifier
```

Description:To be called during calibration each time a calibration kit element is connected and ready to be measured.

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.

C/C++ example:

```
/* Start a full two port calibration for port one and two */
if(( status = ANVNA_StartCalibration(sessionId, "CH1:", true, true, false, false,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT)) != VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 1, ANVNA_VAL_ANRITSU_VNA_CALDATA_SHORT))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 1, ANVNA_VAL_ANRITSU_VNA_CALDATA_OPEN))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 1, ANVNA_VAL_ANRITSU_VNA_CALDATA_LOAD))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}
```

```
if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 2, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_SHORT))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 2, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_OPEN))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 2, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_LOAD))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

if(( status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_THRU))
!= VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

/* End calibration process */
if(( status = ANVNA_EndCalibration(sessionId, "CH1:")) != VI_SUCCESS)
{
ANVNA_AbortCalibration(sessionId, "CH1:");
exit(1);
}

/* Get calibration type */
ViUInt32 calType;
ANVNA_GetCalibrationType(sessionId, "CH1:", &calType);

/* Get calibration active */
ViBoolean calActive;
ANVNA_GetCalActive(sessionId, "CH1:", &calActive);

/* Set calibration status */
ANVNA_SetCalStatus(sessionId, "CH1:", false);

/* Get calibration status */
ViBoolean calStatus;
ANVNA_GetCalStatus(sessionId, "CH1:", &calStatus);
```

C# example:

```
/* Start a full two port calibration for port one and two */
if ((status = ANVNA.StartCalibration(sessionId, "CH1:", true, true, false, false,
(uint)ANVNA.VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 1, 1,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_SHORT)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 1, 1,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_OPEN)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 1, 1,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_LOAD)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 2, 2,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_SHORT)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 2, 2,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_OPEN)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 2, 2,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_LOAD)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
```

```
    System.Environment.Exit(1);
}

if ((status = ANVNA.CalStoreData(sessionId, "CH1:", 1, 2,
(uint)ANVNA.VAL_ANRITSU_VNA_CALDATA_THRU)) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

/* End calibration process */
if ((status = ANVNA.EndCalibration(sessionId, "CH1:")) != 0)
{
    ANVNA.AbortCalibration(sessionId, "CH1:");
    System.Environment.Exit(1);
}

/* Get calibration type */
uint calType;
ANVNA.GetCalibrationType(sessionId, "CH1:", out calType);

/* Get calibration active */
ushort calActive;
ANVNA.GetCalActive(sessionId, "CH1:", out calActive);

/* Set calibration status */
ANVNA.SetCalStatus(sessionId, "CH1:", 1);

/* Get calibration status */
ushort calStatus;
ANVNA.GetCalStatus(sessionId, "CH1:", out calStatus);
```

Python example:

```
# Start a full two port calibration for port one and two
status = ANVNA_StartCalibration(sessionId, "CH1:", True, True, False, False,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 1, ANVNA_VAL_ANRITSU_VNA_CALDATA_SHORT)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)
status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 1, ANVNA_VAL_ANRITSU_VNA_CALDATA_OPEN)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
```

```
    exit(1)

status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 1, ANVNA_VAL_ANRITSU_VNA_CALDATA_LOAD)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

status = ANVNA_CalStoreData(sessionId, "CH1:", 2, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_SHORT)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

status = ANVNA_CalStoreData(sessionId, "CH1:", 2, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_OPEN)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

status = ANVNA_CalStoreData(sessionId, "CH1:", 2, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_LOAD)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

status = ANVNA_CalStoreData(sessionId, "CH1:", 1, 2, ANVNA_VAL_ANRITSU_VNA_CALDATA_THRU)
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

# End calibration process
status = ANVNA_EndCalibration(sessionId, "CH1:")
if status != 0:
    ANVNA_AbortCalibration(sessionId, "CH1:")
    exit(1)

# Get calibration type
status, calType = ANVNA_GetCalibrationType(sessionId, "CH1:")

# Get calibration active
status, calActive = ANVNA_GetCalActive(sessionId, "CH1:")

# Set calibration status
status = ANVNA_SetCalStatus(sessionId, "CH1:", False)

# Get calibration status
status, calStatus = ANVNA_GetCalStatus(sessionId, "CH1:")
```

MATLAB example:

```
% Start a full two port calibration for port one and two
status = anvna.StartCalibration('CH1:', 1, 1, 0, 0,
anvna.VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 1, 1, anvna.VAL_ANRITSU_VNA_CALDATA_SHORT);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 1, 1, anvna.VAL_ANRITSU_VNA_CALDATA_OPEN);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 1, 1, anvna.VAL_ANRITSU_VNA_CALDATA_LOAD);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 2, 2, anvna.VAL_ANRITSU_VNA_CALDATA_SHORT);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 2, 2, anvna.VAL_ANRITSU_VNA_CALDATA_OPEN);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 2, 2, anvna.VAL_ANRITSU_VNA_CALDATA_LOAD);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

status = anvna.CalStoreData('CH1:', 1, 2, anvna.VAL_ANRITSU_VNA_CALDATA_THRU);
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
```

```
end

% End calibration process
status = anvna.EndCalibration('CH1:');
if status ~= 0
    anvna.AbortCalibration('CH1:')
    exit(1)
end

% Get calibration type
status, calType = anvna.GetCalibrationType('CH1:');

% Get calibration active
status, calActive = anvna.GetCalActive('CH1:');

% Set calibration status
status = anvna.SetCalStatus('CH1:', 0);

% Get calibration status
status, calStatus = anvna.GetCalStatus('CH1:');
```

B-38 ANVNA_CalStoreData

```
ViStatus ANVNA_CalStoreData (
    ViSession vi,
    ViConstString RepCapIdentifier,
    ViInt32 sourcePort,
    ViInt32 destPort,
    ViUInt32 dataType);
```

Description: To be called during calibration each time a calibration kit element is connected and ready to be measured.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
sourcePort	ViInt32	Sets port 1 either active or inactive for the current calibration setup (True = Active)
dataType	ViInt32	Sets port 2 either active or inactive for the current calibration setup (True = Active)
numberOfPointVal	ViUInt32	Sets port 3 either active or inactive for the current calibration setup (True = Active)

dataType

#define ANVNA_VAL_ANRITSU_VNA_CALDATA_SHORT	0
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_OPEN	1
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_LOAD	2
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_THRU	3

Example: See ANVNA_StartCalibration.

B-39 ANVNA_EndCalibration

```
ViStatus ANVNA_EndCalibration (
    ViSess
    ion vi,
    ViConstString RepCapIdentifier);
```

Description: Ends the calibration setup. Will return error if not enough measurements are taken using ANVNA_CalStoreData or if ANVNA_StartCalibration is missing.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.

Examples: see Start Calibration Examples.

B-40 ANVNA_AbortCalibration

```
ViStatus ANVNA_AbortCalibration (
    ViSession vi,
    ViConstString RepCapIdentifier);
```

Description: Aborts a calibration in process

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.

C/C++ Examples:

Examples: see Start Calibration Examples.

B-41 ANVNA_GetCalibrationType

```
ViStatus ANVNA_GetCalibrationType (
    ViSession vi,
    ViConstString RepCapIdentifier,
    ViPUInt32 calType);
```

Description: Gets the type of current calibration. Has a defaults value and is gets updated once ANVNA_StartCalibration is called.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
calType	ViPUInt32	The type of calibration value, returned by the function

calType	values:
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT	0
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTFORWARD	1
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSEONEPORT	2
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRFORWARDPATH	3
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTREVERSE	5
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT	4
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_REFLACTIONBOTHPORT	6
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSETWOPORT	7
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRREVERSEPATH	8
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TRFBITHPATHS	9
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTHREEPORT	10
#define ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLFOURPORT	11

Examples: see Start Calibration Examples.

B-42 ANVNA_GetCalibrationMethod

```
ViStatus ANVNA_GetCalibrationMethod (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 calMethod);
```

Purpose: Gets the method of current calibration. Has a defaults value and is gets updated once ANVNA_SetupCalibration is called.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
calMethod	ViPUInt32	Calibration method value, returned by the function. See possible values below.

calMethod	values:
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLT	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SOLR	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSLT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SSST	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRX	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRL	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_LRM	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TOPSHELF_AUTOCAL	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_TRADITIONAL_AUTOCAL	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_SUPER_LRM	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_METHOD_AUTOBOT_AUTOCAL	11

Examples: see example for ANVNA_StartCalibration.

B-43 ANVNA_GetCalActive

```
ViStatus ANVNA_GetCalActive (
    ViSession vi,
    ViConstString RepCapIdentifier,
    ViPBoolean calActive);
```

Description: Function returns whether a calibration exists or not in the current session.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
calActive	ViPBoolean	Result returned, either TRUE (CAL EXISTS) or FALSE (CAL DOES NOT EXIST)

Examples: see Start Calibration Examples.

B-44 ANVNA_LoadCalKit

```
ViStatus ANVNA_LoadCalKit (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViConstString calKitFile,
  ViUInt32 port);
```

Purpose: Loads a calibration kit from file and associates it with a port number.

Parameter List:

Name	Variable Type	Description
vi	ViSession	he ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
calKitFile	iConstString	File name to be loaded.
port	ViUInt32	Port to be associated with.

Examples: see example for ANVNA_StartCalibration.

B-45 ANVNA_LoadS1PKit

```
ViStatus ANVNA_LoadS1PKit (
   ViSession vi,
   ViConstString RepCapIdentifier,
   ViConstString calKitFile,
   ViUInt32 port,
   ViUInt32 dataType);
```

Purpose: Loads a S1P calibration kit from file and associates it with a port number.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
calKitFile	ViConstString	The S1P file name to be loaded.
port	ViUInt32	Port to be associated with.
dataType	ViUInt32	Data type to be loaded.

dataType	values
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_SHORT	0
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_OPEN	1
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_LOAD	2
#define ANVNA_VAL_ANRITSU_VNA_CALDATA_THRU	3

Examples: see example for ANVNA_StartCalibration.

B-46 ANVNA_SetThru

```
ViStatus ANVNA_SetThru (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViUInt32 srcPort,
  ViUInt32 destPort,
  ViReal64 trueThruLength,
  ViReal64 trueThruImpedance,
  ViReal64 trueThruLineLoss,
  ViReal64 trueThruRefFrequency);
```

Purpose: Set thru properties.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
srcPort	ViUInt32	Thru source port.
destPort	ViUInt32	Thru destination port.
trueThruLengthV	ViReal64	Thru length.
trueThruImpedance	ViReal64	Thru impedance.
trueThruLineLoss	ViReal64	Thru line loss.
trueThruRefFrequency	ViReal64	Thru ref frequency.

Examples: see example for ANVNA_StartCalibration.

B-47 ANVNA_GetThru

```
ViStatus ANVNA_GetThru (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViUInt32 srcPort,
  ViUInt32 destPort,
  ViPReal64 trueThruLength,
  ViPReal64 trueThruImpedance,
  ViPReal64 trueThruLineLoss,
  ViPReal64 trueThruRefFrequency);
```

Purpose: Get thru properties.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
srcPort	ViUInt32	Thru source port.
destPort	ViUInt32	Thru destination port.
trueThruLengthV	ViPReal64	Out thru length.
trueThruImpedance	ViPReal64	Out impedance.
trueThruLineLoss	ViPReal64	Out line loss.
trueThruRefFrequency	ViPReal64	Out ref frequency.

Examples: see example for ANVNA_StartCalibration.

B-48 ViStatus ANVNA_SetReciprocalThru

```
ViStatus ANVNA_SetReciprocalThru (
     ViSession vi,
     ViConstString RepCapIdentifier,
     ViUInt32 srcPort,
     ViUInt32 destPort,
     ViBoolean isThruReciprocal);
```

Purpose: Set reciprocal thru state.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
srcPort	ViUInt32	Thru source port.
destPort	ViUInt32	Thru destination port.
trueThruLength	ViBoolean	Reciprocal thru state.

Examples: see example for ANVNA_StartCalibration.

B-49 ANVNA_GetReciprocalThru

```
ViStatus ANVNA_GetReciprocalThru (
     ViSession vi,
     ViConstString RepCapIdentifier,
     ViUInt32 srcPort,
     ViUInt32 destPort,
     ViPBoolean isThruReciprocal);
```

Purpose: Get reciprocal thru state.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
srcPort	ViUInt32	Thru source port.
destPort	ViUInt32	Thru destination port.
isThruReciprocal	ViPBoolean	Reciprocal thru state.

Examples: see example for ANVNA_StartCalibration.

B-50 ANVNA_GetCalStatus

```
ViStatus ANVNA_GetCalStatus (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViPBoolean calStatus);
```

Purpose: Function returns whether a calibration is active or not in the current session. If no calibration exists then an error code is returned.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. "CH1:")**	ViConstString	The physical or virtual repeated capability identifier.
calStatus	ViPBoolean	Result returned, either TRUE (CAL ACTIVE) or FALSE (CAL INACTIVE)

Examples: see example for ANVNA_StartCalibration.

B-51 ANVNA_SetCalStatus

```
ViStatus ANVNA_SetCalStatus (
  ViSession vi,
  ViConstString RepCapIdentifier,
  ViBoolean calStatus);
```

Purpose: Function sets a calibration either active or inactive in the current session.

Note	Activate only if a calibration has been previously performed. Otherwise ANVNA_SetCalStatus will return error if trying to set to “TRUE”.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier **(Only channel index is parsed, ex. “CH1:”)**	ViConstString	The physical or virtual repeated capability identifier.
calStatus	ViBoolean	Calibration status setter either TRUE (CAL ACTIVE) or FALSE (CAL INACTIVE

Examples: see example for ANVNA_StartCalibration.

B-52 ANVNA_AddSelectedPort

```
ViStatus ANVNA_AddSelectedPort (
  ViSession vi,
  ViUInt16 PortNumber,
  ViUInt16 CurrentEncodedPorts,
  ViPUInt16 NewEncodedPorts);
```

Purpose: Helper function to accumulate ports.

Parameter List:

:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
PortNumber	ViUInt16	Port number to be added.
CurrentEncodedPortsV	ViBoolean	Current value. At first call, or when you want to reset the conceded value you need to pass zero.
NewEncodedPorts	ViPUInt16	Result value obtained by combining the PortNumber and CurrentEncodedPorts variables.

C++ example:

```
ViUInt16 port = 0;
    status = ANVNA_AddSelectedPort(sessionId, 1, port, &port);
```

C# example:

```
ushort port;
ANVNA.AddSelectedPort (sessionId, (ushort)1, port, out port);
```

Python example:

```
port = 0
    status, port = ANVNA_AddSelectedPort (sessionId, 1, port)
```

MATLAB example:

```
port = 0;
status, port = anvna.AddSelectedPort (1, port);
```

B-53 ANVNA_SetAutoCalDevice

```
ViStatus ANVNA_SetAutoCalDevice (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViConstString comPort,
  ViConstString characterizationFile,
  ViUInt16 portLeft,
  ViUInt16 portRight,
  ViBoolean orientation,
  ViBoolean autoSenseOn);
```

Purpose: Helper function to accumulate ports.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
comPort	ViConstString	COM port identifier.
characterizationFile	ViConstString	AutoCal device characterization file.
portLeft	ViUInt16	AutoCal left ports.
portRight	ViUInt16	AutoCal right ports.
orientation	ViBoolean	VI_TRUE for left, VI_FALSE for right.
autoSenseOn	ViBoolean	Enable AutoSense capabilitie.

C++ example:

```
ViUInt16 portLeft = 0;
ViUInt16 portRight = 0;
status = ANVNA_AddSelectedPort(sessionId, 1, 0, &portLeft);
status = ANVNA_AddSelectedPort(sessionId, 2, 0, &portRight);
status = ANVNA_SetAutoCalDevice (sessionId, “CH1:”, “COM1”,
“CharacterizationFile.chf”, portLeft, portRight, VI_TRUE, VI_FALSE);
```

C# example:

```
ushort portLeft;
ushort portRight;
ANVNA.AddSelectedPort (sessionId, (ushort)1, portLeft, out portLeft);
ANVNA.AddSelectedPort (sessionId, (ushort)2, portRight, out portRight);
ANVNA.SetAutoCalDevice(sessionId, “CH1:”, “COM1”, “CharacterizationFile.chf”,
portLeft, portRight, True, False);
```

Python example:

```
portLeft = 0
portRight = 0
status, portLeft = ANVNA_AddSelectedPort (sessionId, 1, portLeft)
status, portRight = ANVNA_AddSelectedPort (sessionId, 2, portRight)
status, port = ANVNA_SetAutoCalDevice(sessionId, ‘CH1:’, ‘COM1’,
‘CharacterizationFile.chf’, portLeft, portRight, True, False)
```

MATLAB example:

```
portLeft = 0;
portRight = 0;
status, portLeft = anvna.AddSelectedPort (1, portLeft);
status, portRight = anvna.AddSelectedPort (2, portRight);
status = anvna.SetAutoCalDevice(sessionId, ‘CH1:’, ‘COM1’,
‘CharacterizationFile.chf’, portLeft, portRight, True, False);
```

B-54 ANVNA_GetAutoCalDevice

```
ViStatus ANVNA_GetAutoCalDevice (
  ViPSession vi,
  ViPConstString repCapIdentifier,
  ViPConstString comPort,
  ViPConstString characterizationFile,
  ViPUInt16 portLeft,
  ViPUInt16 portRight,
  ViPBoolean orientation,
  ViPBoolean autoSenseOn);
```

Purpose: Get Auto Calibration device current parameters.

Parameter List:

Name	Variable Type	Description
Vi	ViPSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViPConstString	The physical or virtual repeated capability identifier.
comPort	ViPConstString	COM port identifier.
characterizationFile	ViPConstString	AutoCal device characterization file.
portLeft	ViPUInt16	AutoCal left ports.
portRight	ViPUInt16	AutoCal right ports.
orientation	ViPBoolean	VI_TRUE for left, VI_FALSE for right.
autoSenseOn	ViPBoolean	Enable AutoSense capabilitie.

C++ example:

```
ViChar comPort[256];
ViChar characterizationFile[256];
ViUInt16 portLeft = 0;
ViUInt16 portRight = 0;
ViPBoolean orientation = VI_FALSE;;
ViPBoolean autoSenseOn = VI_FALSE;
status = ANVNA_GetAutoCalDevice (sessionId, &comPort, &characterizationFile,
&portLeft, &portRight, &orientation, &autoSenseOn);
```

C# example:

```
string comPort;
string characterizationFile;
ushort portLeft;
ushort portRight;
ushort orientation;
ushort autoSenseOn;
ANVNA.GetAutoCalDevice(sessionId, out comPort, out characterizationFile, out
portLeft, out portRight, out orientation, out autoSenseOn);
```

Python example:

```
status, comport, characterizationFile, portLeft, portRight, orientation =
ANVNA_GetAutoCalDevice(sessionId, 'CH1:')
```

MATLAB example:

```
status, comport, characterizationFile, portLeft, portRight, orientation =
anvna.GetAutoCalDevice(sessionId, 'CH1:');
```

B-55 ANVNA_SetSmartCalDevice

```
ViStatus ANVNA_SetSmartCalDevice (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt16 NumPorts,
  ViUInt16 PortA,
  ViUInt16 PortB,
  ViUInt16 PortC,
  ViUInt16 PortD,
  ViBoolean autoSenseOn);
```

Purpose: Set SmartCal device parameters.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
NumPorts	ViUInt16	Number of ports.
PortA	ViUInt16	SmartCal port A ports.
PortB	ViUInt16	SmartCal port B ports.
PortC	ViUInt16	SmartCal port C ports.
PortD	ViUInt16	SmartCal port D ports.
autoSenseOn	ViBoolean	Enable AutoSense capabilitie.

C++ example:

```
ViUInt16 portA = 0;
ViUInt16 portB = 0;
status = ANVNA_AddSelectedPort(sessionId, 1, 0, &portA);
status = ANVNA_AddSelectedPort(sessionId, 2, 0, &portB);
status = ANVNA_SetSmartCalDevice (sessionId, 2, portA, portB, 0, 0, VI_TRUE);
```

C# example:

```
ushort portA;
ushort portB;
```

```
ANVNA.AddSelectedPort (sessionId, (ushort)1, portA, out portA);
ANVNA.AddSelectedPort (sessionId, (ushort)2, portB, out portB);
ANVNA.SetSmartCalDevice(sessionId, 2, portA, portB, 0, 0, True);
```

Python example:

```
portA = 0
portB = 0
status, portA = ANVNA_AddSelectedPort (sessionId, 1, portA)
status, portB = ANVNA_AddSelectedPort (sessionId, 2, portB)
status, port = ANVNA_SetSmartCalDevice(sessionId, 2, portA, portB, 0, 0, True)
```

MATLAB example:

```
portA = 0;
portB = 0;
status, portA = anvna.AddSelectedPort (1, portA);
status, portB = anvna.AddSelectedPort (2, portB);
status = anvna.SetSmartCalDevice(sessionId, 2, portA, portB, 0, 0, True);
```

B-56 ANVNA_GetSmartCalDevice

```
ViStatus ANVNA_GetSmartCalDevice (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt16 NumPorts,
  ViPUInt16 PortA,
  ViPUInt16 PortB,
  ViPUInt16 PortC,
  ViPUInt16 PortD,
  ViPBoolean autoSenseOn);
```

Purpose: Get SmartCal device parameters.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
NumPorts	ViPUInt16	Number of ports.
PortA	ViPUInt16	SmartCal port A ports.
PortB	ViPUInt16	SmartCal port B ports.
PortC	ViPUInt16	SmartCal port C ports.
PortD	ViPUInt16	SmartCal port D ports.
autoSenseOn	ViPBoolean	Enable AutoSense capabilitie.

C++ example:

```
ViUInt16 NumPorts = 0;
ViUInt16 portA = 0;
ViUInt16 portB = 0;
ViUInt16 portC = 0;
ViUInt16 portD = 0;
ViPBoolean autoSenseOn = VI_FALSE;
status = ANVNA_GetSmartCalDevice (sessionId, &NumPorts, &portA,  &portB, &portC,
&portD, &autoSenseOn);
```

C# example:

```
ushort NumPorts;
ushort portA;
ushort portB;
ushort portC;
ushort portD;
ushort autoSenseOn;
ANVNA.GetSmartCalDevice(sessionId, out NumPorts, out portA, out portB, out portC,
out portD, out autoSenseOn);
```

Python example:

```
status, NumPorts, portA, portB, portC, portD, autoSenseOn =
ANVNA_GetSmartCalDevice(sessionId, 'CH1:')
```

MATLAB example:

```
status, NumPorts, portA, portB, portC, portD, autoSenseOn =
anvna.GetSmartCalDevice(sessionId, 'CH1:');
```

B-57 ANVNA_SetAdditionalTrue

```
ViStatus ANVNA_SetAdditionalTrue (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt16 portA,
  ViUInt16 portB);
```

Purpose: Set additional thru between VNA portA and VNA portB.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
portA	ViUInt16	VNA port 1.
portB	ViUInt16	VNA port 2.

C++ example:

```
status = ANVNA_SetAdditionalTrue (sessionId, “CH1:”, 1, 2);
```

C# example:

```
ANVNA.SetAdditionalTrue(sessionId, “CH1:”, 1, ,2);
```

Python example:

```
status = ANVNA_SetAdditionalTrue(sessionId, ‘CH1:’, 1, 2)
```

MATLAB example:

```
status = anvna.SetAdditionalTrue(sessionId, ‘CH1:’, 1, 2);
```

B-58 ANVNA_AddOnePortConnection

```
ViStatus ANVNA_AddOnePortConnection (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt16 CalibrationType,
  ViUInt32 portA,
  ViBoolean resetAccumulation);
```

Purpose: Setup device for one port connection.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
CalibrationType	ViUInt16	Calibration type, see calibration types.
portA	ViUInt16	VNA port A.
resetAccumulation	ViBoolean	Reset VNA port accumulation.

CalibrationType	possible values:
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT	0
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTFORWARD	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSEONEPORT	2
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRFORWARDPATH	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTREVERSE	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_REFLECTIONBOTHPORT	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSETWOPORT	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRREVERSEPATH	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TRFBOTHPATHS	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTHREEPORT	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLFOURPORT	1

C++ example:

```
ViUInt16 portA = 0;
status = ANVNA_AddSelectedPort(sessionId, 1, 0, &portA);
status = ANVNA_AddOnePortConnection(sessionId, “CH1:”,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, VI_TRUE);
```

C# example:

```
ushort portA;
ANVNA.AddSelectedPort (sessionId, (ushort)1, portA, out portA);
ANVNA.AddOnePortConnection(sessionId,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, True);
```

Python example:

```
portA = 0
status, portA = ANVNA_AddSelectedPort (sessionId, 1, portA)
status = ANVNA_AddOnePortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, True)
```

MATLAB example:

```
portA = 0;
status, portA = anvna.AddSelectedPort (1, portA);
status = anvna.AddOnePortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, True);
```

B-59 ANVNA_AddTwoPortConnection

```
ViStatus ANVNA_AddTwoPortConnection (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt16 CalibrationType,
  ViUInt32 portA,
  ViUInt32 portB,
  ViBoolean resetAccumulation);
```

Purpose: Setup device for two port connection.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
CalibrationType	ViUInt16	Calibration type, see calibration types.
portA	ViUInt16.	VNA port A
portB	ViUInt16	VNA port B.
resetAccumulation	ViBoolean	Reset VNA port accumulation.

CalibrationType	possible values:
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT	0
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTFORWARD	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSEONEPORT	2
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRFORWARDPATH	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTREVERSE	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_REFLECTIONBOTHPORT	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSETWOPORT	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRREVERSEPATH	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TRFBOTHPATHS	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTHREEPORT	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLFOURPORT	11

C++ example:

```
ViUInt16 portA = 0;
ViUInt16 portB = 0;
status = ANVNA_AddSelectedPort(sessionId, 1, 0, &portA);
status = ANVNA_AddSelectedPort(sessionId, 2, 0, &portB);
status = ANVNA_AddTwoPortConnection(sessionId, “CH1:”,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, VI_TRUE);
```

C# example:

```
ushort portA;
ushort portB;
ANVNA.AddSelectedPort (sessionId, (ushort)1, portA, out portA);
ANVNA.AddSelectedPort (sessionId, (ushort)2, portB, out portB);
ANVNA.AddTwoPortConnection(sessionId,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, True);
```

Python example:

```
portA = 0
portB = 0
status, portA = ANVNA_AddSelectedPort (sessionId, 1, portA)
status, portB = ANVNA_AddSelectedPort (sessionId, 2, portB)
status = ANVNA_AddTwoPortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, True)
```

MATLAB example:

```
portA = 0;
portB = 0;
status, portA = anvna.AddSelectedPort (1, portA);
status, portB = anvna.AddSelectedPort (2, portB);
status = anvna.AddTwoPortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, True);
```

B-60 ANVNA_AddThreePortConnection

```
ViStatus ANVNA_AddThreePortConnection (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt16 CalibrationType,
  ViUInt32 portA,
  ViUInt32 portB,
  ViUInt32 portC,
  ViBoolean resetAccumulation);
```

Purpose: Setup device for three port connection.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
CalibrationType	ViUInt16	Calibration type, see calibration types.
portA	ViUInt16	VNA port A.
portB	ViUInt16	VNA port B.
portC	ViUInt16	VNA port C.
resetAccumulation	ViBoolean	Reset VNA port accumulation.

CalibrationType	possible values:
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT	0
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTFORWARD	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSEONEPORT	2
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRFORWARDPATH	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTREVERSE	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_REFLECTIONBOTHPORT	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSETWOPORT	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRREVERSEPATH	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TRFBOTHPATHS	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTHREEPORT	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLFOURPORT	11

C++ example:

```
ViUInt16 portA = 0;
ViUInt16 portB = 0;
ViUInt16 portC = 0;
status = ANVNA_AddSelectedPort(sessionId, 1, 0, &portA);
status = ANVNA_AddSelectedPort(sessionId, 2, 0, &portB);
status = ANVNA_AddSelectedPort(sessionId, 3, 0, &portC);
status = ANVNA_AddThreePortConnection(sessionId, “CH1:”,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, VI_TRUE);
```

C# example:

```
ushort portA;
ushort portB;
ushort portC;
ANVNA.AddSelectedPort (sessionId, (ushort)1, portA, out portA);
ANVNA.AddSelectedPort (sessionId, (ushort)2, portB, out portB);
ANVNA.AddSelectedPort (sessionId, (ushort)3, portC, out portC);
ANVNA.AddThreePortConnection(sessionId,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, True);
```

Python example:

```
portA = 0
portB = 0
portC = 0
status, portA = ANVNA_AddSelectedPort (sessionId, 1, portA)
status, portB = ANVNA_AddSelectedPort (sessionId, 2, portB)
status, portC = ANVNA_AddSelectedPort (sessionId, 3, portC)
status = ANVNA_AddThreePortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, True)
```

MATLAB example:

```
portA = 0;
portB = 0;
portC = 0;
```

```
status, portA = anvna.AddSelectedPort (1, portA);
status, portB = anvna.AddSelectedPort (2, portB);
status, portC = anvna.AddSelectedPort (3, portC);
status = anvna.AddThreePortConnection(sessionId, 'CH1:',
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC,
```

B-61 ANVNA_AddFourPortConnection

```
ViStatus ANVNA_AddFourPortConnection (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt16 CalibrationType,
  ViUInt32 portA,
  ViUInt32 portB,
  ViUInt32 portC,
  ViUInt32 portD,
  ViBoolean resetAccumulation);
```

Purpose: Setup device for four port connection.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
CalibrationType	ViUInt16	Calibration type, see calibration types.
portA	ViUInt16V	NA port A.
portB	ViUInt16	VNA port B.
portC	ViUInt16	VNA port C.
portD	ViUInt16	VNA port D.
resetAccumulation	ViBoolean	Reset VNA port accumulation.

CalibrationType	possible values:
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT	0
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTFORWARD	1
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSEONEPORT	2
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRFORWARDPATH	3
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTWOPORT	4
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_ONEPATHTWOPORTREVERSE	5
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_REFLECTIONBOTHPORT	6
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_RESPONSETWOPORT	7
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TFRREVERSEPATH	8
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_TRFBOTHPATHS	9
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLTHREEPORT	10
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLFOURPORT	11

C++ example:

```
ViUInt16 portA = 0;
ViUInt16 portB = 0;
ViUInt16 portC = 0;
ViUInt16 portD = 0;
status = ANVNA_AddSelectedPort(sessionId, 1, 0, &portA);
status = ANVNA_AddSelectedPort(sessionId, 2, 0, &portB);
status = ANVNA_AddSelectedPort(sessionId, 3, 0, &portC);
status = ANVNA_AddSelectedPort(sessionId, 4, 0, &portD);
status = ANVNA_AddFourPortConnection(sessionId, “CH1:”,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, portD,
VI_TRUE);
```

C# example:

```
ushort portA;
ushort portB;
ushort portC;
ANVNA.AddSelectedPort (sessionId, (ushort)1, portA, out portA);
ANVNA.AddSelectedPort (sessionId, (ushort)2, portB, out portB);
ANVNA.AddSelectedPort (sessionId, (ushort)3, portC, out portC);
ANVNA.AddSelectedPort (sessionId, (ushort)4, portD, out portD);
ANVNA.AddFourPortConnection(sessionId,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, portD,
True);
```

Python example:

```
portA = 0
portB = 0
portC = 0
status, portA = ANVNA_AddSelectedPort (sessionId, 1, portA)
status, portB = ANVNA_AddSelectedPort (sessionId, 2, portB)
status, portC = ANVNA_AddSelectedPort (sessionId, 3, portC)
status, portD = ANVNA_AddSelectedPort (sessionId, 4, portD)
```

```
status = ANVNA_AddFourPortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, portD,
True)
```

MATLAB example:

```
portA = 0;
portB = 0;
portC = 0;
status, portA = anvna.AddSelectedPort (1, portA);
status, portB = anvna.AddSelectedPort (2, portB);
status, portC = anvna.AddSelectedPort (3, portC);
status, portD = anvna.AddSelectedPort (4, portD);
status = anvna.AddFourPortConnection(sessionId, ‘CH1:’,
ANVNA_VAL_ANRITSU_VNA_CALIBRATION_TYPE_FULLONEPORT, portA, portB, portC, portD,
True);
```

B-62 ANVNA_BeginAutoCalCalibration

```
ViStatus ANVNA_BeginAutoCalCalibration (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPBoolean endStatus,
  ViPUInt16 portA,
  ViPUInt16 portB);
```

Purpose: Start auto calibration process.

:

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
endStatus	ViPBoolean	Is set to VI_TRUE is calibration has ended.
portA	ViPUInt16	VNA port A to be connected.
portB	ViPUInt16	VNA port B to be connected.

C++ example:

```
ViBoolean endStatus = VI_FALSE;
ViUInt16 portA = 0;
ViUInt16 portB = 0;
status = ANVNA_BeginAutoCalCalibration(sessionId, “CH1:”, &endStatus, &portA,
&portB);
```

C# example:

```
ushort endStatus;
ushort portA;
ushort portB;
ANVNA.BeginAutoCalCalibration(sessionId, out endStatus, out portA, out portB);
```

Python example:

```
status, endStatus, portA, portB = ANVNA_BeginAutoCalCalibration(sessionId, ‘CH1:’)
```

MATLAB example:

```
status, endStatus, portA, portB = anvna.BeginAutoCalCalibration(sessionId, ‘CH1:’);
```

B-63 ANVNA_ResumeAutoCalCalibration

```
ViStatus ANVNA_ResumeAutoCalCalibration (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPBoolean endStatus,
  ViPUInt16 portA,
  ViPUInt16 portB);
```

Purpose: Resume calibration process.

:

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
endStatus	ViPBoolean	Is set to VI_TRUE is calibration has ended.
portA	ViPUInt16	VNA port A to be connected.
portB	ViPUInt16	VNA port B to be connected.

C++ example:

```
ViBoolean endStatus = VI_FALSE;
ViUInt16 portA = 0;
ViUInt16 portB = 0;
status = ANVNA_ResumeAutoCalCalibration(sessionId, “CH1:”, &endStatus, &portA,
&portB);
```

C# example:

```
ushort endStatus;
ushort portA;
ushort portB;
ANVNA.ResumeAutoCalCalibration(sessionId, out endStatus, out portA, out portB);
```

Python example:

```
status, endStatus, portA, portB = ANVNA_ResumeAutoCalCalibration(sessionId, ‘CH1:’)
```

MATLAB example:

```
status, endStatus, portA, portB = anvna.ResumeAutoCalCalibration(sessionId,
‘CH1:’);
```

B-64 ANVNA_EndAutoCalCalibration

```
ViStatus ANVNA_EndAutoCalCalibration (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPBoolean endStatus);
```

Purpose: Returnes the calibration process end status.

:

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
endStatus	ViPBoolean	Is set to VI_TRUE is calibration has ended.

C++ example:

```
ViBoolean endStatus = VI_FALSE;
ViUInt16 portA = 0;
ViUInt16 portB = 0;
status = ANVNA_EndAutoCalCalibration(sessionId, “CH1:”, &endStatus, &portA,
&portB);
```

C# example:

```
ushort endStatus;
ushort portA;
ushort portB;
ANVNA.EndAutoCalCalibration(sessionId, out endStatus, out portA, out portB);
```

Python example:

```
status, endStatus, portA, portB = ANVNA_EndAutoCalCalibration(sessionId, ‘CH1:’)
```

MATLAB example:

```
status, endStatus, portA, portB = anvna.EndAutoCalCalibration(sessionId, 'CH1:');
```

B-65 ANVNA_ChannelStimulusRangeConfigureCenterSpan

```
ViStatus ANVNA_ChannelStimulusRangeConfigureCenterSpan (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViReal64 centerVal,
  ViReal64 spanVal);
```

Description: Sets the sweep range for the channel using center and span value of the frequencies.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.
centerVal	ViReal64	Center frequency value, in Hz.
spanVal	ViReal64	Span value, in Hz.

C++ example:

```
    /*
     * Initialisation code ...
     * Measurement setup ...
     */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

if(( status = ANVNA_ChannelStimulusRangeConfigureCenterSpan(sessionId, "CH1:", 4000000000.0,
6000000000.0)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while configuring center and span bandwidth: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// Now CH1 is configured as needed
```

C# example:

```
    /*
     * Initialisation code ...
     * Measurement setup ...
     */
int status = 0;
if ((status = ANVNA.ChannelStimulusRangeConfigureCenterSpan(sessionId, "CH1:", 4000000000.0,
6000000000.0)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while configuring center and span
bandwidth: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
// Now CH1 is configured as needed
```

Python example:

```
    #Initialisation code ...
    #Measurement setup ...

    status = ANVNA_ChannelStimulusRangeConfigureCenterSpan(sessionId,
"CH1:",2000000000,1000000000)
    if status == 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while configuring center and span bandwidth: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Now CH1 is configured as needed
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...

        status = anvna.ChannelStimulusRangeConfigureCenterSpan('CH1:', 4000000000.0,
6000000000.0);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while configuring center and span bandwidth: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;

        end

%   Now CH1 is configured as needed
```

B-66 ANVNA_ChannelStimulusRangeConfigureStartStop

```
ViStatus ANVNA_ChannelStimulusRangeConfigureStartStop (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViReal64 startVal,
  ViReal64 stopVal);
```

Description: Sets the sweep range for the channel using start and stop values of the frequencies.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.
startVal	ViReal64	Start frequency value, in Hz.
stopVal	ViReal64	Stop frequency value, in Hz.

C++ example:

```
    /*
 * Initialisation code ...
 * Measurement setup ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];

if(( status = ANVNA_ChannelStimulusRangeConfigureStartStop(sessionId, "CH1:", 2000000000.0,
6000000000.0)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while configuring start and stop bandwidth: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// Now CH1 is configured as needed
```

C# example:

```
/*
             * Initialisation code ...
             * Measurement setup ...
             */
int status = 0;
if(( status = ANVNA.ChannelStimulusRangeConfigureStartStop(sessionId, "CH1:", 2000000000.0,
6000000000.0)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while configuring start and stop
bandwidth: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
// Now CH1 is configured as needed
```

Python example:

```
    #Initialisation code...
    #Measurement setup...

    status = ANVNA_ChannelStimulusRangeConfigureStartStop(sessionId,
"CH1:",float(2000000000),float(3000000000))
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while configuring start and stop bandwidth: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Now CH1 is configured as needed
```

MATLAB example:

```
%   Initialisation code ...
%   Measurement setup ...

        status = anvna.ChannelStimulusRangeConfigureStartStop('CH1:', 2000000000.0,
6000000000.0);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while configuring start and stop bandwidth: %s\n', stat,
ErrorMessage);
            status = anvna.close();
return;
        end

%   Now CH1 is configured as needed
```

B-67 ANVNA_ChannelTriggerSweep

```
ViStatus ANVNA_ChannelTriggerSweep (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 timeoutVal);
```

Description: This will trigger the sweep on a channel. Method does not return until sweep completes or timeout value is exceeded.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.
timeoutVal	ViInt32	Time out value, in milliseconds.

C++ example:

```
/*
 * Initialisation code ...
 * Trigger sweep ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
// trigger sweep on channel 1 with 2 seconds timeout:
if ((status = ANVNA_ChannelTriggerSweep(sessionId, "CH1:", 2000)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while triggering sweep on channel 1: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
// Sweep done, data can be fetched now …
```

C# example:

```
            /*
             * Initialisation code ...
             * Trigger sweep ...
             */

            int status = 0;
            status = ANVNA.ChannelTriggerSweep(sessionId, "CH1:", 2000);
            if (status != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while triggering sweep on channel
1: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            // Sweep done, data can be fetched now …
```

Python example:

```
    #Initialisation code...
    #Trigger sweep...

    #trigger sweep on channel 1 with 2 seconds timeout:
    status = ANVNA_ChannelTriggerSweep(sessionId, "CH1:", 2000)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while triggering sweep on channel 1: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Sweep done, data can be fetched now...
```

MATLAB example:

```
%   Initialisation code ...
%   Trigger sweep ...

    status = anvna.ChannelTriggerSweep('CH1:',2000);
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while triggering sweep on channel 1: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end

%   Sweep done, data can be fetched now …
```

B-68 ANVNA_close

```
ViStatus ANVNA_close (
  ViSession vi);
```

Description: Closes the connection to the instrument. All subsequent function calls will return error.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.

C++ example:

```
    #  include "ANVNA.h" // Specific driver header file
void main()
{
ViStatus status;
ViSession sessionId;
// Initialize the driver
// This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:
status = ANVNA_Init("TCPIP::127.0.0.1::SOCKET", VI_TRUE, VI_TRUE, & sessionId);

if(status != VI_SUCCESS)
{
printf("Error %d while initializing\n", status);
exit(1);
}

// ... call driver functions using " sessionId " variable

status = ANVNA_close(sessionId);
```

C# example:

```
            int status = 0;
            uint sessionId = 0;
            string ResourceName = "TCPIP:: 127.0.0.1::SOCKET";
            string OptionsString = "Simulate=false";
            // Initialize the driver in non-simulated mode
            // This will also check the instrument ID to make sure it is supported and will
pre-set the instrument to its default state:

            status = ANVNA.init(ResourceName, TRUE, TRUE, out sessionId);
            if (status != 0)
            {
                System.Console.WriteLine("Error " + status + " while initializing");
       System.Environment.Exit(1);
            }

            // ... call driver functions using "sessionID" variable
            status = ANVNA.close(sessionId);
```

Python example:

```
    #Initialize the driver
    #This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:

    status, sessionId = ANVNA_InitWithOptions('TCPIP::127.0.0.1::SOCKET', False, False,
"simulate=false")
    if status != 0:
        print ("Error {0} while initializing".format(status))
        exit(1)

    #call driver functions using " sessionId " variable
    status = ANVNA_close(sessionId)
```

MATLAB example:

```
%   Initialize the driver
%   This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:
    status = anvna.init('TCPIP::127.0.0.1::SOCKET', 1, 1);
    if( status ~= 0)

            fprintf('Error %d while initializing\n', status);
return;

    end

%   ... call driver functions
    status = anvna.close();
```

B-69 ANVNA_error_message

```
ViStatus ANVNA_error_message (
  ViSession vi,
  ViStatus errorCode,
  ViChar errorMessage[]);
```

Description: Translates the error return value from an IVI driver function to a user-readable string. The user should pass a buffer with at least 256 bytes for the ErrorMessage parameter.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
errorCode	ViStatus	Instrument driver status code.
errorMessage	ViChar[]	Output buffer containing the instrument driver error message.
errorMessageLength	ViINt32[]	Size of the output buffer.

C++ example:

```
      /*
 * Initialisation code ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
// trigger sweep on channel 1 with 2 seconds timeout:
if ((status = ANVNA_ChannelTriggerSweep(sessionId, "CH1:", 2000)) != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while triggering sweep on channel 1: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
```

C# example:

```
  /*
 * Initialisation code ...
 * Trigger sweep ...
 * Read attribute's value ANVNA_ATTR_CHANNEL_POINTS into "points"
 */

            int status = 0;
            status = ANVNA.ChannelTriggerSweep(sessionId, "CH1:", 2000);
            if (status != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while triggering sweep on channel
1: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            // Sweep done, data can be fetched now …
```

Python example:

```
    #Initialisation code...
    #Trigger sweep...

    #trigger sweep on channel 1 with 2 seconds timeout:
    status = ANVNA_ChannelTriggerSweep(sessionId, "CH1:", 2000)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while triggering sweep on channel 1: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Sweep done, data can be fetched now...
```

MATLAB example:

```
%   Initialisation code ..

%   trigger sweep on channel 1 with 2 seconds timeout:
    status = anvna.ChannelTriggerSweep('CH1:',2000);
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status); % function is used here
            fprintf('Error %d while triggering sweep on channel 1: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end
```

B-70 ANVNA_GetAttributeViBoolean

```
ViStatus ANVNA_GetAttributeViBoolean (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViBoolean *attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViBoolean* (passed by pointer)	Returns the current value of the attribute. The user must specify the address of a variable that has the same data type as the attribute.

B-71 ANVNA_GetAttributeViInt32

```
ViStatus ANVNA_GetAttributeViInt32 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViInt32 *attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViInt32* (passed by pointer)	Returns the current value of the attribute. The user must specify the address of a variable that has the same data type as the attribute.

Examples: see Generic Attribute Getter examples.

B-72 ANVNA_GetAttributeViInt64

```
ViStatus ANVNA_GetAttributeViInt64 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViInt64 *attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViInt64* (passed by pointer)	Returns the current value of the attribute. The user must specify the address of a variable that has the same data type as the attribute.

Examples: see Generic Attribute Getter examples.

B-73 ANVNA_GetAttributeViReal64

```
ViStatus ANVNA_GetAttributeViReal64 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViReal64 *attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViReal64 (passed by pointer)	Returns the current value of the attribute. The user must specify the address of a variable that has the same data type as the attribute.

Examples: see Generic Attribute Getter examples.

B-74 ANVNA_GetAttributeViString

```
ViStatus ANVNA_GetAttributeViString (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViInt32 attributeValueBufferSize,
  ViChar attributeValue[]);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValueBufferSize	ViInt32	The number of bytes in the ViChar array that the user specifies for the AttributeValue parameter.
attributeValue	ViChar[]	The buffer in which the function returns the current value of the attribute.

Examples: see Generic Attribute Getter examples.

B-75 ANVNA_GetAttributeViUInt32

```
ViStatus ANVNA_GetAttributeViUInt32 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViUInt32 *attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViUInt32* (passed by pointer)	Returns the current value of the attribute. The user must specify the address of a variable that has the same data type as the attribute.

Examples: see Generic Attribute Getter examples.

B-76 ANVNA_GetChannelMeasurementName

```
ViStatus ANVNA_GetChannelMeasurementName (
  ViSession vi,
  ViConstString channel,
  ViInt32 index,
  ViInt32 nameBufferSize,
  ViChar name[]);
```

Description: This function returns the physical identifier that corresponds to the one-based index measurement that the user specifies. If the value that the user passes for the Index parameter is less than one or greater than the value of the corresponding Measurement Count attribute, the function returns an empty string in the Name parameter and returns an error.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	Identifies the instrument session.
channel	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”.
index	ViInt32	A one-based index that defines the trace whose name is to be returned.
nameBufferSize	ViInt32	The number of bytes in the ViChar array that the user specifies for the name parameter.
name	ViChar[]	The buffer into which the function returns the measurement name thatcorresponds to the index the user specifies.

C++ example:

```
        /*
    * Initialisation code ...
    */
ViChar name[MAX_STRING_LENGTH];
if(( status = ANVNA_GetChannelMeasurementName(sessionId, "CH1:", 1, MAX_STRING_LENGTH, name))
!= VI_SUCCESS)
{
ViChar ErrorMessage[MAX_STRING_LENGTH];
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while getting measurement name: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
getchar();
exit(1);
}
printf("Measurement name is %s\n", name);
```

C# example:

```
            string name = "";
            if ((status = ANVNA.GetChannelMeasurementName(sessionId, "CH1:", 1,
ANVNA.MAX_STRING_LENGTH, out name)) != 0)
            {
                string ErrorMessage = "";
    ANVNA.error_message(sessionId, status, out ErrorMessage, ANVNA.MAX_STRING_LENGTH);
    System.Console.WriteLine("Error " + status + " while getting measurement name: " +
ErrorMessage);
    ANVNA.close(sessionId);
                System.Console.Read();
                Environment.Exit(0);
    }

            System.Console.WriteLine("Measurement name is " + name);
```

Python example:

```
    #Initialisation code
    status, name = ANVNA_GetChannelMeasurementName(sessionId, "CH1:", 1, MAX_STRING_LENGTH)
    if status == 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while getting measurement name: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    print ('Measurement name is ' + str(name))
```

MATLAB example:

```
%    * Initialisation code ...

        [status, name] = anvna.GetChannelMeasurementName('CH1:', 1, 1024);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while getting measurement name: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

        fprintf('Measurement name is %s\n', name);
```

B-77 ANVNA_GetChannelName

```
ViStatus ANVNA_GetChannelName (
  ViSession vi,
  ViInt32 index,
  ViInt32 nameBufferSize,
  ViChar name[]);
```

Description: This function returns the physical identifier that corresponds to theone-based index that the user specifies. If the value that the user passes for the Index parameter is less than one or greater than the value of the corresponding Channel Count attribute, the function returns an empty string in the Name parameter and returns an error.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	Identifies the instrument session.
index	ViInt32	A one-based index that defines which channel name to return.
nameBufferSize	ViInt32	The number of bytes in the ViChar array that the user specifies for the name parameter.
name	ViChar[]	The buffer into which the function returns the channel name that corresponds to the index the user specifies.

C++ example:

```
        /*
    * Initialisation code ...
    */
ViChar channelName[MAX_STRING_LENGTH];
if(( status = ANVNA_GetChannelName(sessionId, 1, MAX_STRING_LENGTH, channelName)) !=
VI_SUCCESS)
{
ViChar ErrorMessage[MAX_STRING_LENGTH];
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while getting channel name: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
getchar();
exit(1);
}
printf("Channel name is %s\n", channelName);
```

C# example:

```
            string name = "";
           if ((status = ANVNA.GetChannelName(sessionId, 1, ANVNA.MAX_STRING_LENGTH, out name))
!= 0)
            {
                string ErrorMessage = "";
ANVNA.error_message(sessionId, status, out ErrorMessage, ANVNA.MAX_STRING_LENGTH);
System.Console.WriteLine("Error " + status + " while getting channel name: " + ErrorMessage);
ANVNA.close(sessionId);
                System.Console.Read();
                Environment.Exit(0);
}

            System.Console.WriteLine("Channel name is " + name);
```

Python example:

```
    #Initialisation code ...
    status, channelName = ANVNA_GetChannelName(sessionId,1 , MAX_STRING_LENGTH)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print('Error {0} getting channel name\n'.format(status))
        ANVNA_close(sessionId)
        exit(1)

    print ('Channel name is  ' + str(channelName))
```

MATLAB example:

```
%    Initialisation code ...
        [status, channelName] = anvna.GetChannelName(1, 1024);
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while getting measurement name: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

        fprintf('Channel name is %s\n', channelName);
```

B-78 ANVNA_SetChannelSourcePowerLevel

```
ViStatus ANVNA_SetChannelSourcePowerLevel (
   ViSession vi,
   ViConstString RepCapIdentifier,
   ViInt32 PortVal,
   ViReal64 Val);
```

Purpose: This function sets the power value for the input port.

Parameter List:

Name	Variable Type	Description
vi	ViSession	Identifies the instrument session.
repCapIdentifier	ViConstString	ViConstStringIf the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
PortVal	ViInt32	The port number to be changed.
Val	ViReal64	The power value to be set.

B-79 ANVNA_GetChannelSourcePowerLevel

```
ViStatus ANVNA_GetChannelSourcePowerLevel (
   ViSession vi,
   ViConstString RepCapIdentifier,
   ViInt32 PortVal,
   ViReal64 Val);
```

Purpose: This function gets the power value for the input port.

Parameter List:

Name	Variable Type	Description
vi	ViSession	Identifies the instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
PortVal	ViInt32	The port number to be changed.
Val	ViReal64	The output variable for the power value. 17

B-80 ANVNA_init

```
ViStatus ANVNA_init (
  ViRsrc resourceName,
  ViBoolean idQuery,
  ViBoolean reset,
  ViSession *vi);
```

Description: Opens the communication with the instrument. Driver functions that access the instrument are only accessible after ANVNA_init is called. Initialize optionally performs a Reset and queries the instrument to validate the instrument model.

Cmd Parameters:

Name	Variable Type	Description
resourceName	ViRsrc	An IVI logical name or an instrument specific string that identifies the address of the instrument. Example "TCPIP::192.168.1.2::4857::SOCKET".
idQuery	ViBoolean	Specifies whether to verify the ID of the instrument (VI_TRUE triggers instrument verification).
reset	ViBoolean	Specifies whether to reset the instrument (VI_TRUE triggers reset).
vi	ViSession* (passed by pointer)	Unique identifier for an IVI session.

C++ example:

```
#include "ANVNA.h" // Specific driver header file
void main()
{
ViStatus status;
ViSession sessionId;
// Initialize the driver
// This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:
status = ANVNA_Init("TCPIP::127.0.0.1::SOCKET", VI_TRUE, VI_TRUE, &sessionId);

if(status != VI_SUCCESS)
{
printf("Error %d while initializing\n", status);
exit(1);
}
// ... call driver functions using "sessionID" variable
status = ANVNA_close(sessionId);
}
```

C# example:

```
            int status = 0;
            uint sessionId = 0;
            string ResourceName = "TCPIP::127.0.0.1::SOCKET";
            string OptionsString = "Simulate=false";
            // Initialize the driver in non-simulated mode
            // This will also check the instrument ID to make sure it is supported and will
pre-set the instrument to its default state:

            status = ANVNA.init(ResourceName, TRUE, TRUE, out sessionId);
            if (status != 0)
            {
                System.Console.WriteLine("Error " + status + " while initializing");
                System.Environment.Exit(1);
            }

            // ... call driver functions using "sessionID" variable
            status = ANVNA.close(sessionId);
```

Python example:

```
    #Initialize the driver
    #This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:


    status, sessionId = ANVNA_init('TCPIP::127.0.0.1::SOCKET', True, True)
    if status != 0:
        print("Error {0} while initializing\n".format(status))
        exit(1)

    #call driver functions using "session" variable
```

MATLAB example:

```
    anvna = ANVNADriver(); %Specific driver header file

%   Initialize the driver
%   This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:
    status = anvna.init('TCPIP::127.0.0.1::SOCKET', 1, 1);
    if( status ~= 0)

            fprintf('Error %d while initializing\n', status);
return;
    end

%   call driver functions

    status = anvna.close();
```

B-81 ANVNA_InitWithOptions

```
ViStatus ANVNA_InitWithOptions (
  ViRsrc resourceName
  ViBoolean idQuery,
  ViBoolean reset,
  ViConstString optionsString,
  ViSession *vi);
```

Description: Opens the communication with the instrument. Driver methods and properties that access the instrument are only accessible after Initialize is called. Initialize optionally performs a Reset and queries the instrument to validate the instrument model.

Cmd Parameters:

Name	Variable Type	Description
resourceName	ViRsrc	An IVI logical name or an instrument specific string that identifies the address of the instrument. Example "TCPIP::192.168.1.2::4857::SOCKET".
idQuery	ViBoolean	Specifies whether to verify the ID of the instrument.
reset	ViBoolean	Specifies whether to reset the instrument.
optionsString	ViConstString	The user can use the OptionsString parameter to specify the initial values of certain IVI inherent attributes for the session. The format of an assignment in the OptionsString parameter is "Name=Value", where Name is one of:, Simulate,. Value is either true or false
vi	ViSession* (passed by pointer)	Unique identifier for an IVI session.

C++ example:

```
     #include "ANVNA.h" // Specific driver header file
void main()
{
ViStatus status;
ViSession sessionId;
// Initialize the driver in non-simulated mode
// This will also check the instrument ID to make sure it is supported
status = ANVNA_InitWithOptions("TCPIP::127.0.0.1::SOCKET", VI_TRUE, VI_TRUE, "Simulate=false",
& sessionId);

if(status != VI_SUCCESS)
{
printf("Error %d while initializing\n", status);
exit(1);
}

// ... call driver functions

status = ANVNA_close(sessionId);
}
```

C# example:

```
            int status = 0;
            uint sessionId = 0;
            string ResourceName = "TCPIP::127.0.0.1::SOCKET";
            string OptionsString = "Simulate=false";

            status = ANVNA.InitWithOptions(ResourceName, FALSE, FALSE, OptionsString, out
sessionId);
            if (status != 0)
            {
                System.Console.WriteLine("Error " + status + " while initializing");
                System.Environment.Exit(1);
            }

// ... call driver functions

            status = ANVNA.close(sessionId);
```

Python example:

```
    #Initialize the driver
    #This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:

    status, sessionId = ANVNA_InitWithOptions('TCPIP::127.0.0.1::SOCKET', False, False,
"simulate=false")
    if status != 0:
        print("Error {0} while initializing\n".format(status))
        exit(1)

    #call driver functions using "sessionId" variable

        ANVNA_close(sessionId)
```

MATLAB example:

```
    anvna = ANVNADriver(); %Specific driver header file

%   Initialize the driver in non-simulated mode
%   This will also check the instrument ID to make sure it is supported
    status = anvna.InitWithOptions('TCPIP::127.0.0.1::SOCKET', 1, 1, 'Simulate=false');
    if( status ~= 0)

            fprintf('Error %d while initializing\n', status);
return;
    end

%   ... call driver functions

    status = anvna.close();
```

B-82 ANVNA_reset

```
ViStatus ANVNA_reset (
  ViSession vi);
```

Description: Places the instrument in a known state. For parameter Simulate=True passed by optionsString in ANVNA_InitWithOptions function, ANVNA_reset will NOT change it back to the default value (False).

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.

C++ example:

```
      #include "ANVNA.h" // Specific driver header file
void main()
{
ViStatus status;
ViSession sessionId;
char ErrorMessage[MAX_STRING_LENGTH];

// Initialize the driver in non-simulated mode
// This will also check the instrument ID to make sure it is supported
status = ANVNA_InitWithOptions("TCPIP::127.0.0.1::SOCKET", VI_FALSE, VI_FALSE,
"Simulate=false", & sessionId);

if(status != VI_SUCCESS)
{
printf("Error %d while initializing\n", status);
exit(1);
}

// Reset the instrument
Status = ANVNA_reset(sessionId);
if(status != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while resetting device: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// ... call driver functions

status = ANVNA_close(sessionId);
}
```

C# example:

```
           int status = 0;
            uint sessionId = 0;
status = ANVNA.InitWithOptions("TCPIP::127.0.0.1::SOCKET", FALSE, FALSE, "Simulate=false", out
sessionId);
            if (status != 0)
            {
                System.Console.WriteLine("Error " + status + " while initializing");
                System.Environment.Exit(1);
            }

            // Reset the instrument
            status = ANVNA.reset(sessionId);
            if (status != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while resetting device: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);

            }

            status = ANVNA.close(sessionId);
```

Python example:

```
    #Initialize the driver
    #This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:
    status, sessionId = ANVNA_InitWithOptions('TCPIP::127.0.0.1::SOCKET', False, False,
"simulate=False")
    if status != 0:
        print("Error {0} while initializing\n".format(status))
        exit(1)

    #Reset the instrument
    status = ANVNA_reset(sessionId)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while resetting device: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #call driver functions ...

    status = ANVNA_close(sessionId)
```

MATLAB example:

```
    anvna = ANVNADriver(); %Specific driver header file

%   Initialize the driver in non-simulated mode
%   This will also check the instrument ID to make sure it is supported
    status = anvna.InitWithOptions('TCPIP::127.0.0.1::SOCKET', 0, 0, 'Simulate=false');
    if( status ~= 0)

            fprintf('Error %d while initializing\n', status);
return;

    end

%   Reset the instrument:
    status = anvna.reset();
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while resetting device: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end

%   ... call driver functions

    status = anvna.close();
```

B-83 ANVNA_ResetWithDefaults

```
ViStatus ANVNA_ResetWithDefaults (
  ViSession vi);
```

Description: Does the equivalent of Reset and then configures the driver to option string settings used when ANVNA_InitWithOptions was last executed.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.

C++ Example:

```
#include "ANVNA.h" // Specific driver header file
void main()
{
ViStatus status;
ViSession sessionId;
char ErrorMessage[MAX_STRING_LENGTH];
// Initialize the driver in non-simulated mode
// This will also check the instrument ID to make sure it is supported
status = ANVNA_InitWithOptions("TCPIP::127.0.0.1::SOCKET", VI_FALSE, VI_FALSE,
"Simulate=false", & sessionId);

if(status != VI_SUCCESS)
{
printf("Error %d while initializing\n", status);
exit(1);
}

// Reset the instrument
Status = ANVNA_ResetWithDefaults(sessionId);
if(status != VI_SUCCESS)
{
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while resetting device: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}

// ... call driver functions

status = ANVNA_close(sessionId);
}
```

C# example:

```
            int status = 0;
            uint sessionId = 0;
status = ANVNA.InitWithOptions("TCPIP::127.0.0.1::SOCKET", FALSE, FALSE, "Simulate=false", out
sessionId);
            if (status != 0)
            {
                System.Console.WriteLine("Error " + status + " while initializing");
                System.Environment.Exit(1);
            }

            // Reset the instrument
            status = ANVNA.ResetWithDefaults(sessionId);
            if (status != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while resetting device: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);

            }

            // Call driver functions ...

            status = ANVNA.close(sessionId);
```

Python example:

```
    #Initialize the driver
    #This will also check the instrument ID to make sure it is supported and will pre-set the
instrument to its default state:

    status, sessionId = ANVNA_InitWithOptions('TCPIP::127.0.0.1::SOCKET', False, False,
"simulate=False")
    if status != 0:
        print("Error {0} while initializing\n".format(status))
        exit(1)

    #Reset the instrument
    status = ANVNA_ResetWithDefaults(sessionId)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while resetting device: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    # call driver functions

    status = ANVNA_close(sessionId)
```

MATLAB example:

```
    anvna = ANVNADriver(); %Specific driver header file

%   Initialize the driver in non-simulated mode
%   This will also check the instrument ID to make sure it is supported
    status = anvna.InitWithOptions('TCPIP::127.0.0.1::SOCKET', 0, 0, 'Simulate=false');
    if( status ~= 0)

            fprintf('Error %d while initializing\n', status);
return;

    end

%   Reset the instrument:
    status = anvna.ResetWithDefaults();
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while resetting device: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end

%   ... call driver functions

    status = anvna.close();
```

B-84 ANVNA_revision_query

```
ViStatus ANVNA_revision_query (
  ViSession vi,
  ViChar driverRev[],
  ViChar instrRev[]);
```

Description: Retrieves revision information from the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
driverRev	ViChar[]	Returns the revision of the IVI specific driver, which is the value held in the Specific Driver Revision attribute. Refer to the Specific Driver Revision attribute for more information.
instrRev	ViChar[]	Returns the firmware revision of the instrument, which is the value held in the Instrument Firmware Revision attribute. Refer to the Instrument Firmware Revision attribute for more information.

C++ example:

```
/*
 * Initialisation code ...
 */
ViChar driverRev[MAX_STRING_LENGTH];
ViChar instrRev[MAX_STRING_LENGTH];
if (( status = ANVNA_revision_query(sessionId, driverRev, instrRev)) != VI_SUCCESS)
{
char ErrorMessage[MAX_STRING_LENGTH];
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while retrieving revision information: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
printf("Driver revision is %s and instrument revision is %s\n", driverRev, instrRev);
```

C# example:

```
            string driverRev;
            string instrRev;
            if ((status = ANVNA.revision_query(sessionId, out driverRev, out instrRev)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while retrieving revision
information: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            System.Console.WriteLine("Driver revision is " + driverRev + " and instrument
revision is " + instrRev);
```

Python example:

```
    #Initialisation code ...
    status, driverRev, instrRev = ANVNA_revision_query (sessionId)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while retrieving revision information: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    print("Driver revision is {0} and instrument revision is {1}\n".format(driverRev,
instrRev))
```

MATLAB example:

```
%    * Initialisation code ...

        [status, driverRev, instrRev] = anvna.revision_query();
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
           fprintf('Error %d while retrieving revision information: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

        fprintf('Driver revision is %s and instrument revision is %s\n', driverRev, instrRev);
```

B-85 ANVNA_self_test

```
ViStatus ANVNA_self_test (
  ViSession vi,
  ViInt16 *testResult,
  ViChar testMessage[]);
```

Description: Performs an instrument self test, waits for the instrument to complete the test, and queries the instrument for the results. If the instrument passes the test, TestResult is zero and TestMessage is 'Self test passed'.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
testResult	ViInt16* (passed by pointer)	The numeric result from the self test operation. 0 = no error (test passed)
testMessage	ViChar[]	The self test status message

C++ example:

```
        /*
         * Initialisation code ...
         */
ViInt16 testResult = 0;
ViChar testMessage[MAX_STRING_LENGTH];
if(( status = ANVNA_self_test(sessionId, &testResult, testMessage)) != VI_SUCCESS)
{
char ErrorMessage[MAX_STRING_LENGTH];
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while performing instrument self test: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
printf("Self test result is %d:%s\n", testResult, testMessage);
```

C# example:

```
            string TestMessage = "";
            if ((status = ANVNA.self_test(sessionId, out TestResult, out TestMessage)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while performing instrument self
test: " + ErrorMessage);
                ANVNA.close(sessionId);
                System.Console.Read();
                Environment.Exit(0);
            }

            System.Console.WriteLine("Self test result is " + TestResult.ToString() + ":" +
TestMessage);
```

Python example:

```
status, result, message = ANVNA_self_test (sessionId)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while performing instrument self test: {1}\n".format(status,
ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    print("Self test result is {0}:{1}\n".format(result, message))
```

MATLAB example:

```
%    * Initialisation code ...

        [status, testResult, testMessage] = anvna.self_test();
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
           fprintf('Error %d while performing instrument self test: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

        fprintf('Self test result is %d:%s\n', testResult, testMessage);
```

B-86 ANVNA_SetAttributeViBoolean

```
ViStatus ANVNA_SetAttributeViBoolean (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViBoolean attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViBoolean	The value to which to set the attribute.

Example: See Generic Attribute Setter examples.

B-87 ANVNA_SetAttributeViInt32

```
ViStatus ANVNA_SetAttributeViInt32 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViInt32 attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViInt32	The value to which to set the attribute.

Example: See Generic Attribute Setter examples.

B-88 ANVNA_SetAttributeViInt64

```
ViStatus ANVNA_SetAttributeViInt64 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViInt64 attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViInt64	The value to which to set the attribute.

Example: See Generic Attribute Setter examples.

B-89 ANVNA_SetAttributeViReal64

```
ViStatus ANVNA_SetAttributeViReal64 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViReal64 attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViReal64	The value to which to set the attribute.

Example: See Generic Attribute Setter examples.

B-90 ANVNA_SetAttributeViString

```
ViStatus ANVNA_SetAttributeViString (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViConstString attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViConstString	The value to which to set the attribute.

B-91 ANVNA_SetAttributeViUInt32

```
ViStatus ANVNA_SetAttributeViUInt32 (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViAttr attributeID,
  ViUInt32 attributeValue);
```

Description: This function is used to access low-level settings of the instrument. See the attributeID parameter for a link to all attributes of the instrument.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Otherwise, the user passes VI_NULL or an empty string.
attributeID	ViAttr	The ID of the attribute. The valid values for this parameter can be found in the driver header file.
attributeValue	ViUInt32	The value to which to set the attribute.

Example: See Generic Attribute Setter examples.

B-92 ANVNA_SetMarkerFrequency

```
ViStatus ANVNA_SetMarkerFrequency (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViReal64 frequency);
```

Purpose: Set marker reference frequency. By setting the frequency the marker state is set to on.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
frequency	ViReal64	Input frequency to be set.

Example: TO DO

C++ example:

```
ANVNA_SetMarkerFrequency(sessionId, "CH1:TR1", 1, 1000000000);
ANVNA_SetMarkerState(sessionId, "CH1:TR1", 1, VI_TRUE);

// trigger sweep …

ViReal64 marker_frequency = 0.0;
ViReal64 marker_value = 0.0;

ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1, &marker_frequency);
ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

printf("marker 1 frequency : %f\n", marker_frequency);
printf("marker 1 value : %f\n", marker_value);
```

C# example:

```
ANVNA.SetMarkerFrequency(sessionId, "CH1:TR1", 1, 1000000000);
ANVNA.SetMarkerState(sessionId, "CH1:TR1", 1, (ushort)1);

// trigger sweep …

double marker_value = 0.0;

ANVNA.GetMarkerFrequency(sessionId, "CH1:TR1", 1, out marker_frequency);
ANVNA.GetMarkerValue(sessionId, "CH1:TR1", 1, out marker_value);

System.Console.WriteLine("marker 1 frequency : " + marker_frequency);
System.Console.WriteLine("marker 1 value : %" + marker_value);
```

Python example:

```
status = ANVNA_SetMarkerFrequency(sessionId, "CH1:TR1", 1, float(1000000000));

// trigger sweep …

status, marker_frequency = ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1,
&marker_frequency);
status, marker_value = ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

print('marker 1 frequency : {0}\n'.format(marker_frequency));
print('marker 1 value : {0}\n'.format(marker_value));
```

MATLAB example:

```
status = anvna.SetMarkerFrequency('CH1:TR1', 1, 1000000000);
status = anvna.SetMarkerState('CH1:TR1', 1, 1);

% trigger sweep …

[status, marker_frequency] = anvna.GetMarkerFrequency('CH1:TR1', 1);
[status, marker_value] = anvna.GetMarkerValue('CH1:TR1', 1);
```

```
ANVNA_GetMarkerFrequency
   ViStatus ANVNA_GetMarkerFrequency (
ViSession vi,
ViConstString repCapIdentifier,
   ViUInt32 markerNum,
ViPReal64 frequency);

Purpose
Get marker reference frequency. This function works with inactive markers also.

Parameter List



Name Variable Type Description

vi ViSession The ViSession handle that you obtain from the IviDriver_init or
IviDriver_InitWithOptions function.  The handle identifies a particular instrument session.

repCapIdentifier ViConstString If the attribute applies to a repeated capability, the user
passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.

markerNum ViUInt32 Marker index starting from 1.

frequency ViPReal64 Output frequency.



Example: See Set Marker Frequency
ANVNA_SetMarkerSearch
   ViStatus ANVNA_SetMarkerSearch (
ViSession vi,
ViConstString repCapIdentifier,
   ViUInt32 markerNum,
ViReal64 start,
ViReal64 stop,
ViUInt32 search_type);

Purpose
Setup a search marker on the frequency interval between start and stop looking for MIN or MAX in
that range. By configuring the search marker its state is set to on.
```

Parameter List

Name Variable Type Description

vi ViSession The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.

repCapIdentifier ViConstString If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.

markerNum ViUInt32 Marker index starting from 1.

start ViReal64 Start frequency.

stop ViReal64 Stop frequency.

search_type ViUInt32 Search type is set to minimum or maximum.

Search type values:

#define ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX 0
#define ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MIN 1

C++ example:

```
ANVNA_SetMarkerSearch(sessionId, "CH1:TR1", 1, 1000000000, 2000000000,
ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX);
ANVNA_SetMarkerState(sessionId, "CH1:TR1", 1, VI_TRUE);

// trigger sweep …

ViReal64 start;
ViReal64 stop;
ViUInt32 type;
ANVNA_GetMarkerSearch(sessionId, "CH1:TR1", 1, &start, &stop, &type);

ViBoolean state;
ANVNA_GetMarkerState(sessionId, "CH1:TR1", 1, &state);

ViReal64 marker_frequency = 0.0;
ViReal64 marker_value = 0.0;

ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1, &marker_frequency);
ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

printf("marker 1 frequency : %f\n", marker_frequency);
printf("marker 1 value : %f\n", marker_value);

C# e
```

B-93 ANVNA_StatusPreset

```
ViStatus ANVNA_StatusPreset (
  ViSession vi);
```

Description: Same functionality as ANVNA_Reset.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.

Example: See example for ANVNA_ChannelAsynchronousTriggerSweep.

Generic Attribute Getter Examples

Getter function calls should be performed using the following pattern:

ANVNA_GetAttribute<type>(sessionId, “CH<channel index>:Measurement<trace index>”, <attribute identifier>, <value pointer>);

Example for fetching number of points:

C++ Example:

```
      /*
 * Initialisation code ...
 */
// get number of points:
ViUInt32 points = 0;
if ((status = ANVNA_GetAttributeViUInt32(sessionId, “CH1:”, ANVNA_ATTR_CHANNEL_POINTS,
&points)) != VI_SUCCESS)
{
char ErrorMessage[MAX_STRING_LENGTH];
ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
printf("Error %d while retrieving number of points: %s\n", status, ErrorMessage);
ANVNA_close(sessionId);
exit(1);
}
printf(“number of points is %d\n”, points);
```

C# Example:

```
            /*
       * Initialisation code ...
       */
       // Get the number of points
            uint points;
            if ((status = ANVNA.GetAttributeViUInt32(sessionId, "CH1:",
(uint)ANVNA.ATTR_CHANNEL_POINTS, out points)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
              System.Console.WriteLine("Error " + status + " while retrieving number of points:
" + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            System.Console.WriteLine("number of points " + points);
```

Python Example:

```
    # Initialisation code

    # Get the number of points:
    status, points =  ANVNA_GetAttributeViUInt32 (sessionId, "CH1:Measurement1",
ANVNA_ATTR_CHANNEL_POINTS)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
       print('Error {0} while retrieving number of points: {1}\n'.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    print ("number of points is {0}\n".format(points))
```

MATLAB Example:

```
%    * Initialisation code ...

%   get number of points:
    [status, points] = anvna.GetAttributeViUInt32 ('CH1:', anvna.ATTR_CHANNEL_POINTS);
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while retrieving number of points: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

    end

    fprintf('number of points is %d\n', points);

Generic Attribute Setter Examples


Setter function calls should be performed using the following pattern:
ANVNA_SetAttribute<type>(sessionId, “CH<channel index>:Measurement<trace index>, <attribute
identifier>, <value>);
```

Example for setting number of points:

C++ Example:

```
             /*
         * Initialisation code ...
         */

        // set number of points:
        ViUInt32 points = 501;
        if ((status = ANVNA_SetAttributeViUInt32(sessionId, “CH1:”, ANVNA_ATTR_CHANNEL_POINTS,
points)) != VI_SUCCESS)
            {
                char ErrorMessage[MAX_STRING_LENGTH];
    ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
    printf("Error %d while setting number of points: %s\n", status, ErrorMessage);
    ANVNA_close(sessionId);
    exit(1);
            }
            // Number of points on channel 1 is now 501 …
```

C# Example:

```
             /*
         * Initialisation code ...
         */

        // set number of points:
            uint points = 501;
            if ((status = ANVNA.SetAttributeViUInt32(sessionId, "CH1:",
(uint)ANVNA.ATTR_CHANNEL_POINTS, points)) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while setting number of points:
" + ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }
            // Number of points on channel 1 is now 501 …
```

Python Example:

```
    # Initialisation code ...

    # Set the number of points:
    points = 501
    status =  ANVNA_SetAttributeViUInt32 (sessionId, "CH1:Measurement1",
ANVNA_ATTR_CHANNEL_POINTS, points)
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH);
        print('Error {0} while setting number of points: {1}\n'.format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

   # Number of points on channel 1 is now 501 …
```

MATLAB Example:

```
%    * Initialisation code ...

%   set number of points:
    points=501;
    status = anvna.SetAttributeViUInt32 ('CH1:', anvna.ATTR_CHANNEL_POINTS, points);
    if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while setting number of points: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;
    end
%   Number of points on chan
```

B-94 ANVNA_ChannelRecallState

```
ViStatus ANVNA_ChannelRecallState (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViConstString Identifier);
```

Description: Recalls the instrument state from the specified instrument file for the specified channel. The file will be loaded from the server side for models MS46322A/B, MS46522B, MS46524B and MS46524B and from the PC for MS46121A/B and MS46122A/B.

File extension is used to control save scope:

*.cha: all channel setup and calibration

*.chx: current channel setup and calibration

*.sta: all channel setup

*.stx: current channel setup

Parameter List: C++ example:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”. Used for *.chx and *.stx values to identify the target channel.
Identifier	ViConstString	File name to be loaded for the specified channel.

C++ example:

```
            /*
             * Initialisation code ...
             */
            ViStatus status = VI_SUCCESS;
            if ((status = ANVNA_ChannelRecallState(sessionId, "CH1:",
"C:\\AnritsuVNA\\Cal\\Calibration.chx")) != VI_SUCCESS)
            {
                ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
                printf("Error %d while recalling channel state: %s\n", status, ErrorMessage);
                ANVNA_close(sessionId);
                exit(1);
            }
            // Channel is now in the recalled state …
```

C# example:

```
            /*
             * Initialisation code ...
             */
 int status = 0;
            if ((status = ANVNA.ChannelRecallState(sessionId, "CH1:",
"C:\\AnritsuVNA\\Cal\\Calibration.chx")) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
               System.Console.WriteLine("Error " + status + " while recalling channel state: "
+ ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            // Channel is now in the recalled state …
```

Python example:

```
    #Initialisation code ...
    status = ANVNA_ChannelRecallState (sessionId, "CH1:",
"C:\\AnritsuVNA\\Cal\\Calibration.chx")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while recalling channel state: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Channel is now in the recalled state…
```

MATLAB example:

```
%   Initialisation code ...

        status = anvna.ChannelRecallState ('CH1:', 'C:\\AnritsuVNA\\Cal\\Calibration.chx');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while recalling channel state: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

%   Channel is now in the recalled state …
```

B-95 ANVNA_ChannelMeasurementGetResponseType

ViStatus ANVNA_ChannelMeasurementGetResponseType (
ViSession vi,
ViConstString repCapIdentifier,
ViPUInt32 ResponseType);

:

Purpose: Get the response type.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
ResponseType	ViPUInt32	Response type value.

C++ example:

```
ViStatus status = VI_SUCCESS;
ViUInt32 responseType = 0;
status = ANVNA_ChannelMeasurementGetResponseType(sessionId, “CH1:TR11”,
&responseType);
```

C# example:

```
int status = 0;
uint responseType = 0;
status = ANVNA.ChannelMeasurementGetResponseType(sessionId, “CH1:TR1”, out
responseType);
```

Python example:

```
status, responseType = ANVNA_ChannelMeasurementGetResponseType(sessionId,
“CH1:TR1”)
```

MATLAB example:

```
status, responseType = anvna.ChannelMeasurementGetResponseType('CH1:TR1');
```

B-96 ANVNA_ChannelMeasurementGetResponseType

```
ViStatus ANVNA_ChannelMeasurementGetResponseType (
 ViSession vi,
 ViConstString repCapIdentifier,
 ViPUInt32 ResponseType);
```

:

Purpose: Get the response type.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
ResponseType	ViPUInt32	Response type value.

C++ example:

```
ViStatus status = VI_SUCCESS;
ViUInt32 responseType = 0;
status = ANVNA_ChannelMeasurementGetResponseType(sessionId, “CH1:TR11”,
&responseType);
```

C# example:

```
int status = 0;
uint responseType = 0;
status = ANVNA.ChannelMeasurementGetResponseType(sessionId, “CH1:TR1”, out
responseType);
```

Python example:

```
status, responseType = ANVNA_ChannelMeasurementGetResponseType(sessionId,
“CH1:TR1”)
```

MATLAB example:

```
status, responseType = anvna.ChannelMeasurementGetResponseType('CH1:TR1');
```

B-97 ANVNA_ChannelSaveState

```
ViStatus ANVNA_ChannelSaveState (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViConstString Identifier);
```

Description: Saves the instrument state to the specified instrument file for the specified channel. The saved file will be stored on the box side for models MS46322A/B, MS46522B, MS46524B and MS46524B and on the PC side for MS46121A/B and MS46122A/B models.

*.cha: all channel setup and calibration

*.chx: current channel setup and calibration

*.sta: all channel setup

*.stx: current channel setup

Parameters:List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:”. Used for *.chx and *.stx values to identify the target channel.
identifier	ViConstString	File name to be saved for the specified channel.

C++ example:

```
            /*
             * Initialisation code ...
             */
            ViStatus status = VI_SUCCESS;
            if ((status = ANVNA_ChannelSaveState(sessionId, "CH1:",
"C:\\AnritsuVNA\\Cal\\Calibration.chx")) != VI_SUCCESS)
            {
                ANVNA_error_message(sessionId, status, ErrorMessage, MAX_STRING_LENGTH);
                printf("Error %d while saving channel state: %s\n", status, ErrorMessage);
                ANVNA_close(sessionId);
                exit(1);
            }
            // Channel is now in the recalled state …
```

C# example:

```
            /*
             * Initialisation code ...
             */
 int status = 0;
            if ((status = ANVNA.ChannelSaveState(sessionId, "CH1:",
"C:\\AnritsuVNA\\Cal\\Calibration.chx")) != 0)
            {
                string ErrorMessage = "";
                ANVNA.error_message(sessionId, status, out ErrorMessage,
ANVNA.MAX_STRING_LENGTH);
                System.Console.WriteLine("Error " + status + " while saving channel state: " +
ErrorMessage);
                ANVNA.close(sessionId);
                System.Environment.Exit(1);
            }

            // Channel is now in the recalled state …
```

Python example:

```
    #Initialisation code ...
    status = ANVNA_ChannelSaveState (sessionId, "CH1:", "C:\\AnritsuVNA\\Cal\\Calibration.chx")
    if status != 0:
        ErrorMessage = ANVNA_error_message(sessionId, status, MAX_STRING_LENGTH)
        print("Error {0} while saving channel state: {1}\n".format(status, ErrorMessage))
        ANVNA_close(sessionId)
        exit(1)

    #Channel is now in the recalled state…
```

MATLAB example:

```
%   Initialisation code ...

        status = anvna.ChannelSaveState ('CH1:', 'C:\\AnritsuVNA\\Cal\\Calibration.chx');
        if( status ~= 0)

            [ stat, ErrorMessage ] = anvna.error_message(status);
            fprintf('Error %d while saving channel state: %s\n', stat, ErrorMessage);
            status = anvna.close();
return;

        end

%   Channel is now in the recalled state …
```

B-98 ANVNA_SystemWaitForOperationComplete

```
ViStatus ANVNA_SystemWaitForOperationComplete (
  ViSession vi,
  ViInt32 maxTimeMilliseconds);
```

Description: Function returns when all pending operations are complete or MaxTimeMilliseconds exceeded.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the IviDriver_init or IviDriver_InitWithOptions function. The handle identifies a particular instrument session.
maxTimeMilliseconds	ViInt32	Maximum time out, in milliseconds.

B-99 ANVNA_GetMarkersCount

```
ViStatus ANVNA_GetMarkersCount (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt32 markersCount);
```

:

Description: Function returns the number of all available markers for a trace. The count is done for all active and inactive markers.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markersCount	ViPInt32	Number of available markers.

Example: See example for ANVNA_SetMarkerFrequency function

B-100 ANVNA_SetMarkerFrequency

```
ViStatus ANVNA_SetMarkerFrequency (
    ViSession vi,
    ViConstString repCapIdentifier,
    ViUInt32 markerNum,
    ViReal64 frequency);
```

Description: Set marker reference frequency. By setting its frequency, the marker state is automatically set to on.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
frequency	ViReal64	Input frequency to be set.

C++ example:

```
ANVNA_SetMarkerFrequency(sessionId, "CH1:TR1", 1, 1000000000);

// trigger sweep …

ViReal64 marker_frequency = 0.0;
ViReal64 marker_value = 0.0;

ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1, &marker_frequency);
ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

printf("marker 1 frequency : %f\n", marker_frequency);
printf("marker 1 value : %f\n", marker_value);
```

C# example:

```
ANVNA.SetMarkerFrequency(sessionId, "CH1:TR1", 1, 1000000000);

// trigger sweep …

double marker_value = 0.0;

ANVNA.GetMarkerFrequency(sessionId, "CH1:TR1", 1, out marker_frequency);
ANVNA.GetMarkerValue(sessionId, "CH1:TR1", 1, out marker_value);

System.Console.WriteLine("marker 1 frequency : " + marker_frequency);
System.Console.WriteLine("marker 1 value : %" + marker_value);
```

Python example:

```
status = ANVNA_SetMarkerFrequency(sessionId, "CH1:TR1", 1, float(1000000000));

// trigger sweep …

status, marker_frequency = ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1,
&marker_frequency);
status, marker_value = ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

print('marker 1 frequency : {0}\n'.format(marker_frequency));
print('marker 1 value : {0}\n'.format(marker_value));
```

MATLAB example:

```
status = anvna.SetMarkerFrequency('CH1:TR1', 1, 1000000000);

% trigger sweep …

[status, marker_frequency] = anvna.GetMarkerFrequency('CH1:TR1', 1);
[status, marker_value] = anvna.GetMarkerValue('CH1:TR1', 1);
```

B-101 ANVNA_GetMarkerFrequency

```
ViStatus ANVNA_GetMarkerFrequency (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViReal64 frequency);
```

Description: Get marker reference frequency. This function works with inactive markers also.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
frequency	ViReal64	Output frequency.

Example: See example for ANVNA_SetMarkerFrequency function.

B-102 ANVNA_SetMarkerSearch

```
ViStatus ANVNA_SetMarkerSearch (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViReal64 start,
  ViReal64 stop,
  ViUInt32 search_type);
```

Description: Setup a search marker on the frequency interval between start and stop looking for MIN or MAX in that range. By configuring the search marker its state is set to on.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle ireoces a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
start	ViReal64	Start frequency.
stop	ViReal64	Stop frequency.
search_type	ViUInt32	Search type is set to minimum or maximum.

Search type values:

#define ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX 0

#define ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MIN 1

C++ example:

```
ANVNA_SetMarkerSearch(sessionId, "CH1:TR1", 1, 1000000000, 2000000000,
ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX);
ANVNA_SetMarkerState(sessionId, "CH1:TR1", 1, VI_TRUE);

// trigger sweep …

ViReal64 start;
ViReal64 stop;
ViUInt32 type;
ANVNA_GetMarkerSearch(sessionId, "CH1:TR1", 1, &start, &stop, &type);

ViBoolean state;
ANVNA_GetMarkerState(sessionId, "CH1:TR1", 1, &state);

ViReal64 marker_frequency = 0.0;
ViReal64 marker_value = 0.0;

ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1, &marker_frequency);
ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

printf("marker 1 frequency : %f\n", marker_frequency);
printf("marker 1 value : %f\n", marker_value);
```

C# example:

```
ANVNA.SetMarkerSearch(sessionId, "CH1:TR1", 1, 1000000000, 2000000000,
(uint)ANVNA.VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX);
ANVNA.SetMarkerState(sessionId, "CH1:TR1", 1, (ushort)1);

// trigger sweep …

double start;
double stop;
uint type;
ANVNA.GetMarkerSearch (sessionId, "CH1:TR1", 1, out start, out stop, out type);

ushort state;
ANVNA.GetMarkerState(sessionId, "CH1:TR1", 1, out state);

double marker_frequency = 0.0;
double marker_value = 0.0;

ANVNA.GetMarkerFrequency(sessionId, "CH1:TR1", 1, out marker_frequency);
ANVNA.GetMarkerValue(sessionId, "CH1:TR1", 1, out marker_value);

System.Console.WriteLine("marker 1 frequency : " + marker_frequency);
System.Console.WriteLine("marker 1 value : %" + marker_value);
```

Python example:

```
status = ANVNA_SetMarkerSearch(sessionId, "CH1:TR1", 1, float(1000000000), float(2000000000),
ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX);
status = ANVNA_SetMarkerState(sessionId, "CH1:TR1", 1, True);

// trigger sweep …

status, start, stop, type = ANVNA_GetMarkerSearch(sessionId, "CH1:TR1", 1);
status, state = ANVNA_GetMarkerState(sessionId, "CH1:TR1", 1);

status, marker_frequency = ANVNA_GetMarkerFrequency(sessionId, "CH1:TR1", 1,
&marker_frequency);
status, marker_value = ANVNA_GetMarkerValue(sessionId, "CH1:TR1", 1, &marker_value);

print('marker 1 frequency : {0}\n'.format(marker_frequency));
print('marker 1 value : {0}\n'.format(marker_value));
```

MATLAB example:

```
status = anvna.SetMarkerSearch('CH1:TR1', 1, 1000000000, 2000000000,
anvna.VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX);
status = anvna.SetMarkerState('CH1:TR1', 1, 1);

% trigger sweep …

[status, start, stop, type] = anvna.GetMarkerSearch('CH1:TR1', 1);
[status, state] = anvna.GetMarkerState('CH1:TR1', 1);

[status, marker_frequency] = anvna.GetMarkerFrequency('CH1:TR1', 1);
[status, marker_value] = anvna.GetMarkerValue('CH1:TR1', 1);
```

B-103 ANVNA_GetMarkerSearch

```
ViStatus ANVNA_GetMarkerSearch
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViPReal64 start,
  ViPReal64 stop,
  ViPUInt32 search_type);
```

Description: This function retrieves previously set search marker properties and it works with inactive markers also.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function.The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
start	ViReal64	Start frequency.
stop	ViReal64	Stop frequency.
search_type	ViUInt32	Search type is set to minimum or maximum.

Search type values:

#define ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MAX 0
#define ANVNA_VAL_ANRITSU_VNA_MARKER_SEARCH_TYPE_MIN 1

B-104 Example: See ANVNA_SetMarkerSearch function example.

```
ANVNA_GetMarkerValue
   ViStatus ANVNA_GetMarkerValue (
   ViSession vi,
   ViConstString repCapIdentifier,
   ViUInt32 markerNum,
   ViPReal64 value);
```

Purpose: This function retrieves the data value pointed by the marker. The marker state must be active when calling this function, otherwise an error is returned.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
value	ViPReal64	Marker index starting from 1. Returned marker data value.

Example: See ANVNA_SetMarkerSearch function example

B-105 ANVNA_GetMarkerUpLowValue

```
ViStatus ANVNA_GetMarkerUpLowValue (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViPReal64 up,
  ViPReal64 low);
```

Purpose: This function retrieves the data value pointed by the marker in case of double value trace. The marker state must be active when calling this function, otherwise an error is returned.

Parameters List:

Name	Variable Type	Description

vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function.The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
up	ViPReal64	ViPReal64
low	ViPReal64	ViPReal64

C++ example:

```
ViReal64 marker_up = 0.0;
ViReal64 marker_low = 0.0;

ANVNA_GetMarkerUpLowValue(sessionId, “CH1:TR1”, 1, &marker_up, &marker_low);
```

C# example:

```
double marker_up = 0.0;
double marker_low = 0.0;

ANVNA.GetMarkerUpLowValue(sessionId, “CH1:TR1”, 1, out marker_up, out marker_low);
```

Python example:

```
status, marker_up, marker_low = ANVNA_GetMarkerUpLowValue(sessionId, “CH1:TR1”, 1)
```

MATLAB example:

```
[status, marker_up, marker_low] = anvna.GetMarkerUpLowValue('CH1:TR1', 1);
```

ViStatus ANVNA_GetMarkerState (

B-106 ANVNA_SetMarkerState

```
ViStatus ANVNA_SetMarkerState
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViBoolean on_off);
```

Description: This function activates or deactivates a marker. If the marker is deactivated the marker value is not updated. Trying to fetch the value of an inactive marker will return an error.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function.The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
on_off	ViBoolean	Marker State.

Example: See ANVNA_SetMarkerSearch function example.

B-107 ViStatus ANVNA_GetMarkerState (

```
ViStatus ANVNA_SetMarkerState
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 markerNum,
  ViPBoolean on_off)
```

Description: This function reads the activation state for the input marker number.

Cmd Parameters:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function.The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
markerNum	ViUInt32	Marker index starting from 1.
on_off	ViBoolean	Marker State.

Example: See ANVNA_SetMarkerSearch function example.

B-108 ANVNA_EnableTimeDomainOption

```
ViStatus ANVNA_EnableTimeDomainOption
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViConstString password );
```

Purpose: This function activates time domain option using the given password.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
password	ViConstString	The password to be used for activation.

Example:

C++ example:

```
status = ANVNA_EnableTimeDomainOption(sessionId, ":", "password");
```

C# example:

```
ANVNA.EnableTimeDomainOption(sessionId, ":", "password");
```

Python example:

```
status = ANVNA_EnableTimeDomainOption (sessionId, ":", "password");
```

MATLAB example:

```
status = anvna.EnableTimeDomainOption(':', 'password');
```

B-109 ANVNA_IsTimeDomainInstalled

```
ViStatus ANVNA_IsTimeDomainInstalled (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPBoolean on_off);
```

Purpose: This function checks time domain option.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
on_off	ViBoolean	Install state: true or false.

Example:

C++ example:

```
ViBoolean hasTimeDomain;
ANVNA_IsTimeDomainInstalled(sessionId, ":", &hasTimeDomain);
```

C# example:

```
ushort hasTimeDomain;
ANVNA.IsTimeDomainInstalled(sessionId, ":", out hasTimeDomain);
```

Python example:

```
status, hasTimeDomain =
ANVNA_IsTimeDomainInstalled(sessionId, ":");
```

MATLAB example:

```
[status, hasTimeDomain] = anvna.IsTimeDomainInstalled(':');
```

B-110 ANVNA_SetTimeDomainType

```
ViStatus ANVNA_SetTimeDomainType (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 domainType);
```

Purpose: Set time domain type.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
domainType	ViUInt32	For accepted values see below list.

domainType possible values:
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_FREQUENCYNOTIMEGATE
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_FREQUENCYTIMEGATE
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMEBANDPASS
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMELOWPASS

Example:

C++ example:

```
status = ANVNA_SetTimeDomainType(sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMELOWPASS);
```

C# example:

```
ANVNA.SetTimeDomainType(sessionId, "CH1:",
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMELOWPASS);
```

Python example:

```
status = ANVNA_SetTimeDomainType(sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMELOWPASS);
```

MATLAB example:

```
status = anvna.SetTimeDomainType('CH1:',
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMELOWPASS);
```

B-111 ANVNA_GetTimeDomainType

```
ViStatus ANVNA_GetTimeDomainType (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 domainType);
```

Purpose: Get time domain type.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
domainType	ViPUInt32	For possible return values see below list.

domainType possible values:
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_FREQUENCYNOTIMEGATE
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_FREQUENCYTIMEGATE
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMEBANDPASS
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TYPE_TIMELOWPASS

Example:

C++ example:

```
ViUInt32 domainType;
status = ANVNA_GetTimeDomainType(sessionId, "CH1:",  &domainType);
```

C# example:

```
uint domainType;
ANVNA.GetTimeDomainType(sessionId, "CH1:", out domainType);
```

Python example:

```
Status, domainType = ANVNA_GetTimeDomainType(sessionId, "CH1:");
```

MATLAB example:

```
[status, domainType] = anvna.GetTimeDomainType('CH1:');
```

B-112 ANVNA_SetTimeDomainResponse

```
ViStatus ANVNA_SetTimeDomainResponse
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 response);
```

Purpose: Set time domain response.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
response	ViUInt32	For possible input values see below list.

domainType possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_STEP

Example:

C++ example:

```
status = ANVNA_SetTimeDomainResponse(sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

C# example:

```
ANVNA.SetTimeDomainResponse(sessionId, "CH1:",
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

Python example:

```
status = ANVNA_SetTimeDomainResponse (sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

MATLAB example:

```
status = anvna.SetTimeDomainResponse('CH1:',
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

B-113 ANVNA_GetTimeDomainResponse

```
ViStatus ANVNA_GetTimeDomainResponse
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 response);
```

Purpose: Set time domain type.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
response	ViUInt32	For possible input values see below list.

domainType possible values:

#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE

#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_STEP

Example:

C++ example:

```
ViUInt32 domainResponse;
status = ANVNA_GetTimeDomainResponse(sessionId, "CH1:", &domainResponse);
```

C# example:

```
uint domainResponse;
ANVNA.GetTimeDomainResponse(sessionId, "CH1:", out domainResponse);
```

Python example:

```
status, domainResponse = ANVNA_GetTimeDomainResponse(sessionId, "CH1:");
```

MATLAB example:

```
[status, domainResponse] = anvna.GetTimeDomainResponse('CH1:');
```

B-114 ANVNA_SetTimeDomainTrip

```
ViStatus ANVNA_SetTimeDomainTrip
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 tripMode)
```

Purpose: Get time domain trip.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
tripMode	ViUInt32	For possible return values see below list.

tripMode possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TRIP_AUTO
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TRIP_ONEWAY
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TRIP_ROUNDTRIP

Example:

C++ example:

```
status = ANVNA_SetTimeDomainResponse(sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

C# example:

```
ANVNA.SetTimeDomainResponse(sessionId, "CH1:",
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

Python example:

```
status = ANVNA_SetTimeDomainResponse (sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

MATLAB example:

```
status = anvna.SetTimeDomainResponse('CH1:',
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_RESPONSE_IMPULSE);
```

B-115 ANVNA_GetTimeDomainTrip

```
ViStatus ANVNA_GetTimeDomainTrip (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 tripMode);
```

Purpose: Get time domain trip.

Parameters List:

Name	Variable Type	Description
vi	ViSession	he ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
tripMode	ViPUInt32	For possible return values see below list.

tripMode possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TRIP_AUTO
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TRIP_ONEWAY
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_TRIP_ROUNDTRIP

Example:

C++ example:

```
ViUInt32 domainTrip;
status = ANVNA_GetTimeDomainTrip(sessionId, "CH1:", &domainTrip);
```

C# example:

```
uint domainTrip;
ANVNA.GetTimeDomainTrip(sessionId, "CH1:", out domainTrip);
```

Python example:

```
status, domainTrip = ANVNA_GetTimeDomainTrip(sessionId, "CH1:");
```

MATLAB example:

```
[status, domainTrip] = anvna.GetTimeDomainTrip('CH1:');
```

B-116 ANVNA_SetTimeDomainUnit

```
ViStatus ANVNA_SetTimeDomainUnit
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 unit);
```

Purpose: Set time domain unit.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
unit	ViUInt32	For possible return values see below list.

unit possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_TIME
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_DISTANCE

Example:

C++ example:

```
status = ANVNA_SetTimeDomainUnit(sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_TIME);
```

C# example:

```
ANVNA.SetTimeDomainUnit(sessionId, "CH1:",
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_TIME);
```

Python example:

```
status = ANVNA_SetTimeDomainUnit(sessionId, "CH1:",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_TIME);
```

MATLAB example:

```
status = anvna.SetTimeDomainUnit('CH1:', anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_TIME);
```

B-117 ANVNA_GetTimeDomainUnit

```
ViStatus ANVNA_GetTimeDomainTrip (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 unit);
```

Purpose: Get time domain trip.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	ViConstStringIf the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
unit	ViPUInt32	For possible return values see below list.

Unit possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_TIME
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_UNIT_DISTANCE

Example:

C++ example:

```
ViUInt32 domainUnit;
status = ANVNA_GetTimeDomainUnit(sessionId, "CH1:", &domainUnit);
```

C# example:

```
uint domainUnit;
ANVNA.GetTimeDomainUnit(sessionId, "CH1:", out domainUnit);
```

Python example:

```
status, domainUnit = ANVNA_GetTimeDomainUnit(sessionId, "CH1:");
```

MATLAB example:

```
[status, domainUnit] = anvna.GetTimeDomainUnit('CH1:');
```

B-118 ANVNA_SetTimeDomainRangeStartStop

```
ViStatus ANVNA_SetTimeDomainRangeStartStop (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViReal64 start,
  ViReal64 stop);
```

Purpose: Set time domain range using start and stop values.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
start	ViReal64	Range start value.
stop	ViReal64	Range stop value.

Example:

C++ example:

```
status = ANVNA_SetTimeDomainRangeStartStop(sessionId, "CH1:TR1", 50000000, 60000000);
```

C# example:

```
ANVNA.SetTimeDomainRangeStartStop(sessionId, "CH1:TR1", 50000000, 60000000);
```

Python example:

```
status = ANVNA_SetTimeDomainRangeStartStop(sessionId, "CH1:TR1", 50000000, 60000000);
```

MATLAB example:

```
status = anvna.SetTimeDomainRangeStartStop('CH1:TR1', 50000000, 60000000);
```

B-119 ANVNA_GetTimeDomainRangeStartStop

```
ViStatus ANVNA_GetTimeDomainRangeStartStop (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPReal64 start,
  ViPReal64 stop);
```

Purpose: Get time domain range using start and stop values.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	ViConstStringIf the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
start	ViPReal64	Range start value.
stop	ViPReal64	Range stop value.

Example:

C++ example:

```
ViReal64 start, stop;
status = ANVNA_GetTimeDomainRangeStartStop(sessionId, "CH1:TR1", &start,  &stop);
```

C# example:

```
double start, stop;
ANVNA.GetTimeDomainRangeStartStop(sessionId, "CH1:TR1", out start, out stop);
```

Python example:

```
status, start, stop = ANVNA_GetTimeDomainRangeStartStop(sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, start, stop] = anvna.GetTimeDomainRangeStartStop('CH1:TR1');
```

B-120 ANVNA_SetTimeDomainRangeCenterSpan

```
ViStatus ANVNA_SetTimeDomainRangeCenterSpan (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViReal64 center,
  ViReal64 span);
```

Purpose: Set time domain range center and span values.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
Center	ViPReal64	Range center value.
span	ViPReal64	Range span value.

Example:

C++ example:

```
ViReal64 center, span;status = ANVNA_GetTimeDomainRangeCenterSpan(sessionId,
"CH1:TR1", &center,  &span);
```

C# example:

```
double center, span;ANVNA.GetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1", out
center, out span);
```

Python example:

```
status, center, span = ANVNA_GetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, center, span] = anvna.GetTimeDomainRangeStartStop('CH1:TR1');
```

B-121 ANVNA_GetTimeDomainRangeCenterSpan

```
ViStatus ANVNA_GetTimeDomainRangeCenterSpan (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPReal64 center,
  ViPReal64 span);
```

Purpose: Get time domain range center and span values.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
Center	ViPReal64	Range center value.
span	ViPReal64	Range span value.

Example:

C++ example:

```
ViReal64 center, span;
status = ANVNA_GetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1", &center,  &span);
```

C# example:

```
double center, span;
ANVNA.GetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1", out center, out span);
```

Python example:

```
status, center, span = ANVNA_GetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, center, span] = anvna.GetTimeDomainRangeStartStop('CH1:TR1');
```

B-122 ANVNA_SetTimeDomainRangeDCTerm

```
ViStatus ANVNA_SetTimeDomainRangeDCTerm (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 extrapolationType,
  ViReal64 extrapolation,
  ViUInt32 extrapolationMethod);
```

Purpose: Set time domain DC Term values.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
extrapolationType	ViUInt32	For extrapolation types see below list.
extrapolation	ViReal64	Extrapolation value.
extrapolationMethod	ViUInt32	For extrapolation methods see below list.

extrapolationType possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_AUTOEXTRAPOLATE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_LINEIMPEDANCE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_OPEN
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_SHORT
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_OTHER

extrapolationMethod possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_PHASEONLY
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_MAGPHASE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_USERDEFINED

Example:

C++ example:

```
status = ANVNA_SetTimeDomainRangeDCTerm(sessionId, "CH1:TR1",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_AUTOEXTRAPOLATE, 0.0,
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_PHASEONLY);
```

C# example:

```
ANVNA.SetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1", ANVNA.
VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_AUTOEXTRAPOLATE, 0.0, ANVNA.
VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_PHASEONLY);
```

Python example:

```
status = ANVNA_SetTimeDomainRangeCenterSpan(sessionId, "CH1:TR1",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_AUTOEXTRAPOLATE, 0.0,
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_PHASEONLY);
```

MATLAB example:

```
status = anvna.SetTimeDomainRangeCenterSpan('CH1:TR1',
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_AUTOEXTRAPOLATE, 0.0,
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_PHASEONLY);
```

B-123 ANVNA_GetTimeDomainRangeDCTerm

```
ViStatus ANVNA_SetTimeDomainRangeDCTerm (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 extrapolationType,
  ViPReal64 extrapolation,
  ViPUInt32 extrapolationMethod,
  ViPReal64 reflectionCoefficient);
```

Purpose: Get time domain DC Term values.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
extrapolationType	ViUInt32	For extrapolation types see below list.
extrapolation	ViReal64	Extrapolation value.
extrapolationMethod	ViUInt32	For extrapolation methods see below list.
reflectionCoefficient	ViPReal64	Gets the reflection coefficient value.

extrapolationType possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_AUTOEXTRAPOLATE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_LINEIMPEDANCE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_OPEN
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_SHORT
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_OTHER

extrapolationMethod possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_PHASEONLY
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_MAGPHASE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_DCTERM_METHOD_USERDEFINED

Example:

C++ example:

```
ViUInt32 extrapolationType, extrapolationMethod;
ViReal64 extrapolation, reflectionCoefficient;
status = ANVNA_GetTimeDomainRangeDCTerm(sessionId, "CH1:TR1", &extrapolationType,
&extrapolation, &extrapolationMethod, &reflectionCoefficient);
```

C# example:

```
uint extrapolationType, extrapolationMethod;
double extrapolation, reflectionCoefficient;
ANVNA.GetTimeDomainRangeDCTerm(sessionId, "CH1:TR1", out  extrapolationType,
out extrapolation, out extrapolationMethod, out reflectionCoefficient);
```

Python example:

```
status, extrapolationType, extrapolation, extrapolationMethod,
reflectionCoefficient = ANVNA_GetTimeDomainRangeDCTerm (sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, extrapolationType, extrapolation, extrapolationMethod,
reflectionCoefficient] = anvna.GetTimeDomainRangeDCTerm('CH1:TR1');
```

B-124 ANVNA_SetTimeDomainRangeProperties

```
ViStatus ANVNA_SetTimeDomainRangeProperties
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 windowShape,
  ViReal64 shapeValue,
  ViReal64 aliasFreeRange);
```

Purpose: Set time domain range properties: windows shape, shape value and alias free range.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
windowShape	ViUInt32	For extrapolation types see below list.
shapeValue	ViReal64	Custom shape value.
aliasFreeRange	ViReal64	Alias free range value.

windowShape possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_NOMINAL
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_RECTANGULAR
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_LOWSIDELOBE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_MINSIDELOBE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_KAISERBESSEL
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_DOLPHCHEBYSHEV

Example:

C++ example:

```
status = ANVNA_SetTimeDomainRangeProperties(sessionId, "CH1:TR1",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_NOMINAL, 0.0, 0.0);
```

C# example:

```
ANVNA.SetTimeDomainRangeProperties(sessionId, "CH1:TR1",
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_NOMINAL, 0.0, 0.0);
```

Python example:

```
status = ANVNA_SetTimeDomainRangeProperties(sessionId, "CH1:TR1",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_NOMINAL, 0.0, 0.0);
```

MATLAB example:

```
status = anvna.SetTimeDomainRangeProperties('CH1:TR1',
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_NOMINAL, 0.0,  0.0);
```

B-125 ANVNA_GetTimeDomainRangeProperties

```
ViStatus ANVNA_GetTimeDomainRangeProperties
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 windowShape,
  ViPReal64 shapeValue);
```

Purpose: Get time domain range properties: windows shape, shape value.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”
windowShape	ViPUInt32	For extrapolation types see below list.
shapeValue	ViPReal64	Custom shape value.

windowShape possible values:
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_NOMINAL
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_RECTANGULAR
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_LOWSIDELOBE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_MINSIDELOBE
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_KAISERBESSEL
#define ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_WINDOWSHAPE_DOLPHCHEBYSHEV

Example:

C++ example:

```
ViUInt32 windowShape;
ViReal64 shapeValue;
status = ANVNA_GetTimeDomainRangeProperties(sessionId, "CH1:TR1", &extrapolationType,
&shapeValue);
```

C# example:

```
uint windowShape;
double shapeValue;
ANVNA.GetTimeDomainRangeProperties(sessionId, "CH1:TR1", out  windowShape, out
shapeValue);
```

Python example:

```
status, windowShape, shapeValue = ANVNA_GetTimeDomainRangeProperties(sessionId,
"CH1:TR1");
```

MATLAB example:

```
[status, windowShape, shapeValue] = anvna.GetTimeDomainRangeProperties('CH1:TR1');
```

B-126 ANVNA_SetTimeDomainGateStartStop

```
ViStatus ANVNA_SetTimeDomainGateStartStop (
  ViConstString RepCapIdentifier,
  ViReal64 start,
  ViReal64 stop);
```

Purpose: Set time domain gate start and stop interval.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	ViConstStringIf the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
start	ViReal64	For extrapolation types see below list.
stop	ViReal64	Custom shape value.

Example:

C++ example:

```
status = ANVNA_SetTimeDomainGateStartStop(sessionId, "CH1:TR1", 50000000, 60000000);
```

C# example:

```
ANVNA.SetTimeDomainGateStartStop(sessionId, "CH1:TR1", 50000000, 60000000);
```

Python example:

```
status = ANVNA_SetTimeDomainGateStartStop(sessionId, "CH1:TR1", 50000000, 60000000);
```

MATLAB example:

```
status = anvna.SetTimeDomainGateStartStop('CH1:TR1', 50000000, 60000000);
```

B-127 ANVNA_GetTimeDomainGateStartStop

```
ViStatus ANVNA_GetTimeDomainGateStartStop (
  ViConstString RepCapIdentifier,
  ViReal64 start,
  ViReal64 stop);
```

Purpose: Get time domain gate start and stop interval.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
start	ViReal64	Interval start value.
stop	ViReal64	Interval stop value.

Example:

C++ example:

```
ViReal64 start, stop;
status = ANVNA_GetTimeDomainGateStartStop(sessionId, "CH1:TR1", &start,  &stop);
```

C# example:

```
double start, stop;
ANVNA.GetTimeDomainGateStartStop(sessionId, "CH1:TR1", out start, out stop);
```

Python example:

```
status, start, stop = ANVNA_GetTimeDomainGateStartStop(sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, start, stop] = anvna.GetTimeDomainGateStartStop('CH1:TR1');
```

B-128 ANVNA_SetTimeDomainGateCenterSpan

```
ViStatus ANVNA_SetTimeDomainGateCenterSpan (
  ViConstString RepCapIdentifier,
  ViReal64 center,
  ViReal64 span);
```

Purpose: Set time domain gate center and span interval.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
center	ViReal64	Center interval.
span	ViReal64	Interval span.

Example:

C++ example:

```
status = ANVNA_SetTimeDomainGateCenterSpan(sessionId, "CH1:TR1", 50000000, 10000000);
```

C# example:

```
ANVNA.SetTimeDomainGateCenterSpan(sessionId, "CH1:TR1", 50000000, 10000000);
```

Python example:

```
status = ANVNA_SetTimeDomainGateCenterSpan(sessionId, "CH1:TR1", 50000000, 10000000);
```

MATLAB example:

```
status = anvna.SetTimeDomainGateCenterSpan('CH1:TR1', 50000000, 10000000);
```

B-129 ANVNA_GetTimeDomainGateCenterSpan

```
ViStatus ANVNA_GetTimeDomainGateCenterSpan (
  ViConstString RepCapIdentifier,
  ViPReal64 center,
  ViPReal64 span);
```

Purpose: Get time domain gate center and span interval.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
center	ViPReal64	Center interval.
span	ViPReal64	Interval span.

Example:

C++ example:

```
ViReal64 center, span;
status = ANVNA_GetTimeDomainGateCenterSpan(sessionId, "CH1:TR1", &center,  &span);
```

C# example:

```
double center, span;
ANVNA.GetTimeDomainGateCenterSpan(sessionId, "CH1:TR1", out center, out span);
```

Python example:

```
status, center, span = ANVNA_GetTimeDomainGateCenterSpan(sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, center, span] = anvna.GetTimeDomainGateStartStop('CH1:TR1');
```

B-130 ANVNA_SetTimeDomainGateProperties

```
ViStatus ANVNA_SetTimeDomainGateProperties (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 setup,
  ViBoolean notch,
  ViUInt32 shape,
  ViReal64 shapeValue);
```

Purpose: Set time domain gate properties: setup, notch, shape and shape value.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
setup	ViUInt32	Please see possible values in the below list.
notch	ViBoolean	If notch is activated or not.
shape	ViUInt32	Gate shape type.
shapevalue	ViReal64	If shape is KAISERBESSEL or DOLPHCHEBYSHEV this value considered as shape value.

Setup possible values:
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_DISPLAY
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_OFF
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_ON

Shape possible values:
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_NOMINAL
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_MINIMUM
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_MAXIMUM
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_WIDE
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_KAISERBESSEL
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_DOLPHCHEBYSHEV

Example:

C++ example:

```
status = ANVNA_SetTimeDomainGateProperties(sessionId, "CH1:TR1",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_ON, VI_FALSE,
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_NOMINAL, 0.0);
```

C# example:

```
ANVNA.SetTimeDomainGateProperties(sessionId, "CH1:TR1",
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_ON, 0,
ANVNA.VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_NOMINAL, 0.0);
```

Python example:

```
status = ANVNA_SetTimeDomainGateProperties(sessionId, "CH1:TR1",
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_ON, False,
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_NOMINAL, 0.0);
```

MATLAB example:

```
status = anvna.SetTimeDomainGateProperties('CH1:TR1',
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_ON, 0,
anvna.VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_NOMINAL, 0.0);
```

B-131 ANVNA_GetTimeDomainGateProperties

```
ViStatus ANVNA_GetTimeDomainGateProperties (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViPUInt32 setup,
  ViPBoolean notch,
  ViPUInt32 shape,
  ViPReal64 shapeValue);
```

Purpose: Get time domain gate properties: setup, notch, shape and shape value.

Parameters List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
setup	ViPUInt32	Please see possible values in the below list.
notch	ViPBoolean	If notch is activated or not.
shape	ViPUInt32	Gate shape type.
shapevalue	ViPReal64	If shape is KAISERBESSEL or DOLPHCHEBYSHEV this value considered as shape value.

Setup possible values:
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_DISPLAY
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_OFF
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATE_ON

Shape possible values:
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_NOMINAL
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_MINIMUM
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_MAXIMUM
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_WIDE
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_KAISERBESSEL
ANVNA_VAL_ANRITSU_VNA_TIMEDOMAIN_GATESHAPE_DOLPHCHEBYSHEV

Example:

C++ example:

```
ViUInt32 setup, shape;
ViBoolean notch;
ViReal64 shapeValue;
status = ANVNA_GetTimeDomainGateProperties(sessionId, "CH1:TR1", &setup, &notch,
&shape, &shapeValue);
```

C# example:

```
uint setup, shape;
double shapeValue;
ANVNA.GetTimeDomainGateProperties(sessionId, "CH1:TR1", out setup, out notch, out
shape, out shapeValue);
```

Python example:

```
status, setup, notch, shape, shapeValue =
ANVNA_GetTimeDomainGateProperties(sessionId, "CH1:TR1");
```

MATLAB example:

```
[status, windowShape, shapeValue] = anvna.GetTimeDomainGateProperties('CH1:TR1');
```

B-132 ANVNA_GetTimeDomainDistanceValues

ViSession Vi,
ViConstString RepCapIdentifier,
ViInt32 RetValBufferSize,
ViPReal64 RetVal,
ViPInt32 RetValActualSize);

Purpose: Get time domain distance values. The OX-axe values of the time domain setup.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier.
RetValBufferSize	ViInt32	User allocated buffer size.
RetVal	ViPReal64	Destination buffer address.
RetValActualSize	ViPInt32	Returned number of elements from buffer.

C++ Example:

```
ViReal64 RetVal[1024];
ViInt32 RetValActualSize;
status = ANVNA_GetTimeDomainDistanceValues(sessionId, “CH1:TR1”, 1024, &RetVal,
&RetValActualSize);
```

C# Example:

```
double RetVal[1024];
int RetValActualSize;
ANVNA.GetTimeDomainDistanceValues(sessionId, “CH1:TR1”, 1024, out RetVal, out
RetValActualSize);
```

Python Example:

```
   status, RetVal, RetValActualSize = ANVNA_GetTimeDomainDistanceValues(sessionId,
“CH1:TR1”, 1024);
```

MATLAB Example:

```
[status, RetVal, RetValActualSize] = anvna.GetTimeDomainDistanceValues('CH1:TR1', 1024);
```

B-133 ANVNA_GetTimeDomainGateValues

```
ViStatus ANVNA_SetLimitTestingOnOff (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViInt32 RetValBufferSize,
  ViPReal32 RetVal,
  ViPInt32 RetValActualSize);
```

Purpose: Get time domain gate values. For each OX value there is a gate value.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.

Name	Variable Type	Description
repCapIdentifier	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
RetValBufferSize	ViInt32	User allocated buffer size.
RetVal	ViPReal32	Destination buffer address.
RetValActualSize	ViPInt32	Returned number of elements from buffer.

C++ Example:

```
ViReal32 RetVal[1024];
ViInt32 RetValActualSize;
status = ANVNA_GetTimeDomainGateValues(sessionId, “CH1:TR1”, 1024, &RetVal,
&RetValActualSize);
```

C# Example:

```
float RetVal[1024];
int RetValActualSize;
ANVNA.GetTimeDomainGateValues(sessionId, “CH1:TR1”, 1024, out RetVal, out
RetValActualSize);
```

Python Example:

```
   status, RetVal, RetValActualSize = ANVNA_GetTimeDomainGateValues(sessionId,
“CH1:TR1”, 1024);
```

MATLAB Example:

```
[status, RetVal, RetValActualSize] = anvna.GetTimeDomainGateValues('CH1:TR1',
1024);
```

B-134 ANVNA_SetLimitTestingOnOff

```
ViStatus ANVNA_SetLimitTestingOnOff (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViBoolean onOff);
```

Purpose: Activates the limit testing if the boolean parameter is set on true of deactivates it otherwise.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
onOff	ViBoolean	Limit testing state.

C++ Example:

```
status = ANVNA_SetLimitTestingOnOff ( sessionId, “CH1:TR1”, VI_TRUE );
```

C# Example:

```
ANVNA.SetLimitTestingOnOff(sessionId, “CH1:TR1”, 1);
```

Python Example:

```
status = ANVNA_SetLimitTestingOnOff(sessionId, 'CH1:TR1', 1)
```

MATLAB Example:

```
status = anvna.SetLimitTestingOnOff('CH1:Trace1', 1);
```

B-135 ANVNA_GetLimitTestingOnOff

```
ViStatus ANVNA_GetLimitTestingOnOff (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPBoolean onOff);
```

Purpose: Returns the state of limit testing query, declared per trace. Default value is 1.

Parameter List:

Name Variable Type	Description	
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
onOff	ViPBoolean	Limit testing state.

C++ Example:

```
ViBoolean limitTestingStatus = VI_FALSE;
status = ANVNA_GetLimitTestingOnOff ( sessionId, “CH1:TR1”, &limitTestingStatus );
```

C# Example:

```
ushort onOff;
      ANVNA.GetLimitTestingOnOff(sessionId, “CH1:TR1”, out onOff);
```

Python Example:

```
status, limitSet = ANVNA_GetLimitTestingOnOff(sessionId, 'CH1:TR1')
    print('Limit testing get value is {0}'.format(limitSet))
```

MATLAB Example:

```
[status, limitTesting] = anvna.GetLimitTestingOnOff('CH1:Trace1');
```

B-136 ANVNA_SetLimitTestResultSign

```
ViStatus ANVNA_SetLimitTestResultSign (
  ViSession vi,
  ViBoolean onOff);
```

Purpose: Activates the limit testing result sign if the boolean parameter is set on true of deactivates it otherwise.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
onOff	ViBoolean	Limit testing sign state.

C++ Example:

```
status = ANVNA_SetLimitTestResultSign ( sessionId, VI_TRUE );
```

C# Example:

```
ANVNA.SetLimitTestResultSign(sessionId, 1);
```

Python Example:

```
status = ANVNA_SetLimitTestResultSign(sessionId, 1)
```

MATLAB Example:

```
status = anvna.SetLimitTestResultSign(1);
```

B-137 ANVNA_GetLimitTestResultSign

```
ViStatus ANVNA_GetLimitTestResultSign (
  ViSession vi,
  ViPBoolean onOff);
```

Purpose: Returns the state of limit testing result sign declared per trace. Default value is 0.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
onOff	ViPBoolean	Limit testing result sign state.

C++ Example:

```
ViBoolean limitTestingSignStatus = VI_FALSE;
status = ANVNA_GetLimitTestResultSign ( sessionId, &limitTestingSignStatus );
```

C# Example:

```
ushort limitSign;
      ANVNA.GetLimitTestResultSign(sessionId, out limitSign);
```

Python Example:

```
status, limitVisible = ANVNA_GetLimitTestResultSign(sessionId)
      print('Limit testing sign get value is {0}'.format(limitVisible))
```

MATLAB Example:

```
[status, limitSignTesting] = anvna.GetLimitTestResultSign();
```

B-138 ANVNA_ClearAllLimits

```
ViStatus ANVNA_ClearAllLimits (
  ViSession vi,
  ViConstString repCapIdentifier);
```

Purpose: Deletes all the limits for a channel trace.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.

C++ example:

```
status = ANVNA_ClearAllLimits ( sessionId, “CH1:TR1” );
```

C# example:

```
ANVNA.ClearAllLimits(sessionId, “CH1:TR1”);
```

Python example:

```
status = ANVNA_ClearAllLimits(sessionId, 'CH1:TR1')
```

MATLAB example:

```
status = anvna.ClearAllLimits('CH1:Trace1');
```

B-139 ANVNA_GetLimitsCount

```
ViStatus ANVNA_GetLimitsCount (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt32 countLimits);
```

Purpose: Returns the number of limits declared per trace. Default value is 0. The function will return 0 if no segments are declared or after ANVNA_ClearAllLimits is called.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstStringThe physical or virtual repeated capability identifier.	
countLimits	ViPUInt32	Number of limits returned by the `function`

C++ Example:

```
ViUInt32 limits = 0;
status = ANVNA_GetLimitsCount ( sessionId, “CH1:TR1”, &limits );
printf(“Number of limits is %d\n”, limits );
```

C# Example:

```
UInt32 limitsCount;
      ANVNA.GetLimitsCount(sessionId, “CH1:TR1”, out limitsCount);
```

Python Example:

```
status, limitsCount = ANVNA_GetLimitsCount(sessionId, 'CH1:TR1')
print('Limit number is {0}'.format(limitsCount))
```

MATLAB Example:

```
[status, noOfLimits] = anvna.GetLimitsCount('CH1:Trace1');
```

B-140 ANVNA_AddLimit

```
ViStatus ANVNA_AddLimit (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViReal64 x1,
  ViReal64 x2,
  ViReal32 y1,
  ViReal32 y2,
  ViReal32 radius,
  ViUInt32 limitType );
```

Purpose: Adds a new limit using Start and Stop frequency, Y1 and Y2, radius – for circular limits and limitType values.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
x1	ViReal64	Start frequency in Hz.
x2	ViReal64	Stop frequency in Hz.
y1	ViReal32	y1 value
y2	ViReal32	y2 value
radius	ViReal32	Radius value for circular limits.
limitType	ViUInt32	Limit type. 0 –ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER for upper limit and 1 - ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER for lower limit.

C++ example:

```
      ViReal64 startFrequency = 3000000000;
ViReal64 stopFrequency = 8000000000;
ViReal32 y1 = 9.8;
ViReal32 y2 = 9.8;
ViReal32 radius = 4.5;
status = ANVNA_AddLimit ( sessionId, “CH1:TR1”, startFrequency, stopFrequency, y1,
y2, radius, ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER );
```

C# example:

```
ANVNA.AddLimit(sessionId, “CH1:TR1”, startFrequency, stopFrequency, (float) -9.8,
(float) 9.8, (float) 4.5, (uint)ANVNA.VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER);
```

Python example:

```
    x1 = float(10000000); x2 = float(8000000000); y1 = -99.8; y2 = -99.8; radius =
float(4.5);
   status = ANVNA_AddLimit(sessionId, 'CH1:TR1', x1, x2, y1, y2, radius,
ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER)
```

MATLAB example:

```
status = anvna.AddLimit('CH1:Trace1', startFrequency, stopFrequency, -9.4, 13.4,
5.5, anvna.VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER);
```

B-141 ANVNA_SetLimit

```
ViStatus ANVNA_SetLimit (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViReal64 x1,
  ViReal64 x2,
  ViReal32 y1,
  ViReal32 y2,
  ViReal32 radius,
  ViUInt32 limitType );
```

Purpose: Set configuration for an existing limit using Start and Stop frequency, Y1 and Y2, radius – for circular limits and limitType values.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
x1	ViReal64	Start frequency in Hz.
x2	ViReal64	Stop frequency in Hz.
y1	ViReal32	y1 value
y2	ViReal32	y2 value
radius	ViReal32	Radius value for circular limits.
limitType	ViUInt32	Limit type. 0 –ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER for upper limit and 1 - ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER for lower limit.

C++ example:

```
      ViReal64 startFrequency = 3000000000;
ViReal64 stopFrequency = 8000000000;
ViReal32 y1 = 9.8;
ViReal32 y2 = 9.8;
ViReal32 radius = 4.5;
```

```
status = ANVNA_SetLimit ( sessionId, “CH1:TR1”, 1, startFrequency, stopFrequency,
y1, y2, radius, ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER );
```

C# example:

```
ANVNA.SetLimit(sessionId, “CH1:TR1”, (uint)1, startFrequency, stopFrequency,
(float)-9.8, (float)9.8, (float)4.5, (uint)ANVNA.VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER);
```

Python example:

```
    x1 = float(10000000); x2 = float(8000000000); y1 = -99.8; y2 = -99.8; radius =
float(4.5);
   status = ANVNA_SetLimit(sessionId, 'CH1:TR1', 1, x1, x2, y1, y2, radius,
ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER)
```

MATLAB example:

```
status = anvna.SetLimit('CH1:Trace1', 1, startFrequency, stopFrequency, -9.4, 13.4,
5.5, anvna.VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER);
```

B-142 ANVNA_GetLimit

```
ViStatus ANVNA_GetLimit (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViPReal64 x1,
  ViPReal64 x2,
  ViPReal32 y1,
  ViPReal32 y2,
  ViPReal32 radius,
  ViPUInt32 limitType );
```

Purpose: Get configuration for an existing limit, Start and Stop frequency, Y1 and Y2, radius – for circular limits and limitType values.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
x1	ViPReal64	Start frequency in Hz.
x2	ViPReal64	Stop frequency in Hz.
y1	ViPReal32	y1 value
y2	ViPReal32	y2 value
radius	ViPReal32	Radius value for circular limits.
limitType	ViPUInt32	Output parameter for limit type.

C++ example:

```
      ViReal64 startFrequency = 0;
ViReal64 stopFrequency = 0;
ViReal32 y1 = 0;
ViReal32 y2 = 0;
ViReal32 radius = 0;
ViUInt32 limitType = 0;
status = ANVNA_GetLimit ( sessionId, “CH1:TR1”, 1, &startFrequency, &stopFrequency,
&y1, &y2, &radius, &limitType );
```

C# example:

```
double outX1, outX2;
      float outY1, outY2, outRadius;
      uint outLimitType;
      ANVNA.GetLimit(sessionId, “CH1:TR1”, 1, out outX1, out outX2, out outY1, out outY2, out outRadius, out outLimitType);
```

Python example:

```
    status, outX1, outX2, outY1, outY2, outRadius, outLimitType = ANVNA_GetLimit(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
[status, outStartFreq, outStopFreq, outY1, outY2, outRadius, outLimitType] = anvna.GetLimit('CH1:Trace1', 1);
```

B-143 ANVNA_SetLimitType

```
ViStatus ANVNA_SetLimitType (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViUInt32 limitType );
```

Purpose: Set limit type for an existing limit using limitType value.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
limitType	ViUInt32	Limit type. 0 –ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER for upper limit and 1 - ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER for lower limit.

C++ example:

```
    status = ANVNA_SetLimitType ( sessionId, “CH1:TR1”, 1,
ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER );
```

C# example:

```
ANVNA.SetLimitType(sessionId, “CH1:TR1”, (uint)1,
(uint)ANVNA.VAL_ANRITSU_VNA_LIMIT_TYPE_UPPER);
```

Python example:

```
    status = ANVNA_SetLimitType(sessionId, 'CH1:TR1', 1,
ANVNA_VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER)
```

MATLAB example:

```
status = anvna.SetLimitType('CH1:TR1', 1, anvna.VAL_ANRITSU_VNA_LIMIT_TYPE_LOWER)
```

B-144 ANVNA_GetLimitType

```
ViStatus ANVNA_GetLimitType (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViPUInt32 limitType );
```

Purpose: Get type for an existing limit using limitType parameter.

Parameter List:

Name	Variable Type	Description
Vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(ex. “CH1:TR1”) ViConstStringThe physical or virtual repeated capability identifier.		
limitSegmentNumber	ViUInt32	Limit index number.
limitType	ViPUInt32	Output paramerer for limit type.

C++ example:

```
      ViUInt32 limitType = 0;
status = ANVNA_GetLimitType ( sessionId, “CH1:TR1”, 1, &limitType );
```

C# example:

```
uint outLimitType1;
ANVNA.GetLimitType(sessionId, “CH1:TR1”, 1, out outLimitType1);
```

Python example:

```
    status, outLimitType = ANVNA_GetLimitType(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
status, outLimitType = anvna.GetLimitType('CH1:Trace1', 1);
```

B-145 ANVNA_DeleteLimitSegmentAt

```
ViStatus ANVNA_DeleteLimitSegmentAt (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 limitSegmentNumber);
```

Purpose: Deletes limit segment at index specified by limitSegmentNumber, for a trace.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.

C++ example:

```
status = ANVNA_DeleteLimitSegmentAt ( sessionId, “CH1:TR1”, 1 );
```

C# example:

```
ANVNA.DeleteLimitSegmentAt(sessionId, “CH1:TR1”, 1);
```

Python example:

```
status = ANVNA_DeleteLimitSegmentAt(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
status = anvna.DeleteLimitSegmentAt('CH1:Trace1', 1);
```

B-146 ANVNA_IsLimitTestPass

```
ViStatus ANVNA_IsLimitTestPass (
ViSession vi,
ViConstString repCapIdentifier,
ViPBoolean passNoPass );
```

Purpose: Checks if the limit test is failing or not for a channel trace.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier(ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
passNoPass	ViBoolean	Limit testing result.

C++ example:

```
ViBoolean limitTestingResultStatus = VI_FALSE;
status = ANVNA_IsLimitTestPass ( sessionId, “CH1:TR1”, &limitTestingResultStatus );
```

C# example:

```
ushort outLimitTestPass;
ANVNA.IsLimitTestPass(sessionId, “CH1:TR1”, out outLimitTestPass);
```

Python example:

```
status, isLimitTestPass = ANVNA_IsLimitTestPass(sessionId, 'CH1:TR1')
```

MATLAB example:

```
[status, limitPass] = anvna.IsLimitTestPass('CH1:Trace1');
```

B-147 ANVNA_GetLowerLimitBuffer

```
ViStatus ANVNA_GetLowerLimitBuffer (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 dataIndexXSize,
  ViPInt32 dataIndexX,
  ViPInt32 dataIndexXActualSize,
  ViInt32 dataValueYSize,
  ViPReal32 dataValueY,
  ViPInt32 dataValueYActualSize );
```

Purpose: Returns frequency index values and y values for lower limit. If multiple lower limits are defined they are mixed together in order to obtain only one lower limit.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataIndexXSize	ViInt32	Number of elements in dataIndexX.
dataIndexX	ViReal32[]	Output buffer containing the frequency index values.
dataIndexXActualSize	ViPInt32	Actual number of elements in dataIndexX.
dataValueYSize	ViInt32	Number of elements in dataValueY.
dataValueY	ViReal32[]	Output buffer containing the lower limit y values.
dataValueYActualSize	ViPInt32	Actual number of elements in dataValueY.

C++ example:

```
     ViInt32 noOfPts = 10;
ViInt32 outXActualSize;
ViInt32 outValueYActualSize;
ViInt32 *dataIndexX = new ViInt32 [10];
ViReal32 *dataValueY = new ViReal32 [10];
status = ANVNA_GetLowerLimitBuffer ( sessionId, “CH1:TR1”, noOfPts, dataIndexX,
&outXActualSize, noOfPts, dataValueY, &outValueYActualSize );
delete [] dataIndexX;
delete [] dataValueY;
```

C# example:

```
int noPoints = 10;
int[] indexX = new int[noPoints];
float[] valueY = new float[noPoints];
int indexXactualSize, valueYactualSize;

ANVNA.GetLowerLimitBuffer(sessionId, “CH1:TR1”, (int)noPoints, indexX, out
indexXactualSize, (int)noPoints, valueY, out valueYactualSize);
```

Python example:

```
status, dataIndexX, dataIndexXactualSize, dataValueY, dataValueYactualSize =
ANVNA_GetLowerLimitBuffer(sessionId, 'CH1:TR1', 3, 3)
```

MATLAB example:

```
[status, dataIndexX, dataIndexXActualSize, dataValueY, dataValueYActualSize] =
anvna.GetLowerLimitBuffer('CH1:Trace1', noPoints, noPoints);
```

B-148 ANVNA_GetUpperLimitBuffer

```
ViStatus ANVNA_GetUpperLimitBuffer (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 dataIndexXSize,
  ViPInt32 dataIndexX,
  ViPInt32 dataIndexXActualSize,
  ViInt32 dataValueYSize,
  ViPReal32 dataValueY,
  ViPInt32 dataValueYActualSize );
```

Purpose: Returns frequency index values and y values for upper limit. If multiple upper limits are defined they are mixed together in order to obtain only one upper limit.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataIndexXSize	ViInt32	Number of elements in dataIndexX.
dataIndexX	ViReal32[]	Output buffer containing the frequency index values.
dataIndexXActualSize	ViPInt32	Actual number of elements in dataIndexX.
dataValueYSize	ViInt32	Number of elements in dataValueY.
dataValueY	ViReal32[]	Output buffer containing the upper limit y values.
dataValueYActualSize	ViPInt32	Actual number of elements in dataValueY.

C++ example:

```
     ViInt32 noOfPts = 10;
ViInt32 outXActualSize;
ViInt32 outValueYActualSize;
ViInt32 *dataIndexX = new ViInt32 [10];
ViReal32 *dataValueY = new ViReal32 [10];
status = ANVNA_ GetUpperLimitBuffer ( sessionId, “CH1:TR1”, noOfPts, dataIndexX,
&outXActualSize, noOfPts, dataValueY, &outValueYActualSize );
delete [] dataIndexX;
delete [] dataValueY;
```

C# example:

```
noPoints = 10;
int[] indexX = new int[noPoints];
float[] valueY = new float[noPoints];
int indexXactualSize, valueYactualSize;

ANVNA.GetUpperLimitBuffer(sessionId, “CH1:TR1”, (int)noPoints, indexX, out
indexXactualSize, (int)noPoints, valueY, out valueYactualSize);
```

Python example:

```
status, dataIndexX, dataIndexXactualSize, dataValueY, dataValueYactualSize = ANVNA_
GetUpperLimitBuffer (sessionId, 'CH1:TR1', 3, 3)
```

MATLAB example:

```
[status, dataIndexX, dataIndexXActualSize, dataValueY, dataValueYActualSize] =
anvna.GetUpperLimitBuffer('CH1:Trace1', noPoints, noPoints);
```

B-149 ANVNA_GetLowerLimitFailPointsBuffer

```
ViStatus ANVNA_GetLowerLimitFailPointsBuffer (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 dataValueXSize,
  ViPReal64 dataValueX,
  ViPInt32 dataValueXActualSize,
  ViInt32 dataValueYSize,
  ViPReal32 dataValueY,
  ViPInt32 dataValueYActualSize,
  ViInt32 dataValueFailedYSize,
  ViPReal32 dataValueFailedY,
  ViPInt32 dataValueFailedYActualSize )
```

Purpose: Returns frequency values, y values, y failed values for lower limit.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataValueXSize	ViInt32	Number of elements in dataValueX.
dataValueX	ViReal64[]	Output buffer containing the frequency values.
dataValueXActualSize	ViPInt32	Actual number of elements in dataValueX.
dataValueYSize	ViInt32	Number of elements in dataValueY.
dataValueY	ViReal32[]	Output buffer containing the lower limit y values.
dataValueYActualSize	ViPInt32	Actual number of elements in dataValueY.
dataValueFailedYSize	ViInt32	Number of elements in dataValueFailedY.
dataValueFailedY	ViReal32[]	Output buffer containing the lower limit failed y values.
dataValueFailedYActual Size	ViPInt32	Actual number of elements in dataValueFailedY.

C++ example:

```
      ViInt32 noOfPts = 10;
  ViInt32 outXActualSize;
  ViInt32 outValueYActualSize;
  ViInt32 outFailedYActualSize;
```

```
ViReal64 *dataValueX = new ViReal64 [10];
ViReal32 *dataValueY = new ViReal32 [10];
ViReal32 *dataFailedValueY = new ViReal32 [10];
status = ANVNA_GetLowerLimitFailPointsBuffer ( sessionId, “CH1:TR1”, noOfPts,
dataValueX, &outXActualSize, noOfPts, dataValueY, &outValueYActualSize, noOfPts,
dataFailedValueY, &outFailedYActualSize );

delete [] dataValueX;
delete [] dataValueY;
delete [] dataFailedValueY;
```

C# example:

```
int noPts = 10;
 double[] valueX = new double[noPts];
 float[] valueY = new float[noPts];
 float[] valueFailedY = new float[noPts];
 int valueXactualSize, valueYactualSize, valueFailedYactualSize;

 ANVNA.GetLowerLimitFailPointsBuffer(sessionId, “CH1:TR1”, noPts, valueX, out
valueXactualSize, noPts, valueY, out valueYactualSize, noPts, valueFailedY, out
valueFailedYactualSize);
```

Python example:

```
status, dataValueX, dataIndexXactualSize, dataValueY, dataValueYactualSize,
dataValueFailedY, dataValueFailedYactualSize =
ANVNA_GetUpperLimitFailPointsBuffer(sessionId, 'CH1:TR1', 4, 4, 4)
```

MATLAB example:

```
[status, dataValueX, dataValueXActualSize, dataValueY, dataValueYActualSize,
dataValueFailedY, dataValueFailedYActualSize] =
anvna.GetLowerLimitFailPointsBuffer('CH1:Trace1', noPoints, noPoints, noPoints);
```

B-150 ANVNA_GetUpperLimitFailPointsBuffer

```
ViStatus ANVNA_GetUpperLimitFailPointsBuffer (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViInt32 dataValueXSize,
  ViPReal64 dataValueX,
  ViPInt32 dataValueXActualSize,
  ViInt32 dataValueYSize,
  ViPReal32 dataValueY,
  ViPInt32 dataValueYActualSize,
  ViInt32 dataValueFailedYSize,
  ViPReal32 dataValueFailedY,
  ViPInt32 dataValueFailedYActualSize );
```

Purpose: Returns frequency values, y values, y failed values for upper limit.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataValueXSize	ViInt32	Number of elements in dataValueX.
dataValueX	ViReal64[]	Output buffer containing the frequency values.
dataValueXActualSize	ViPInt32	Actual number of elements in dataValueX.
dataValueYSize	ViInt32	Number of elements in dataValueY.
dataValueY	ViReal32[]	Output buffer containing the lower limit y values.
dataValueYActualSize	ViPInt32	Actual number of elements in dataValueY.
dataValueFailedYSize	ViInt32	Number of elements in dataValueFailedY.
dataValueFailedY	ViReal32[]	Output buffer containing the lower limit failed y values.
dataValueFailedYActual Size	ViPInt32	Actual number of elements in dataValueFailedY.

C++ example:

```
      ViInt32 noOfPts = 10;
  ViInt32 outXActualSize;
  ViInt32 outValueYActualSize;
  ViInt32 outFailedYActualSize;
```

```
ViReal64 *dataValueX = new ViReal64 [10];
ViReal32 *dataValueY = new ViReal32 [10];
ViReal32 *dataFailedValueY = new ViReal32 [10];
status = ANVNA_GetUpperLimitFailPointsBuffer ( sessionId, “CH1:TR1”, noOfPts,
dataValueX, &outXActualSize, noOfPts, dataValueY, &outValueYActualSize, noOfPts,
dataFailedValueY, &outFailedYActualSize );

delete [] dataValueX;
delete [] dataValueY;
delete [] dataFailedValueY;
```

C# example:

```
int noPts = 10;
double[] valueX = new double[noPts];
float[] valueY = new float[noPts];
float[] valueFailedY = new float[noPts];
int valueXactualSize, valueYactualSize, valueFailedYactualSize;

ANVNA.GetUpperLimitFailPointsBuffer(sessionId, “CH1:TR1”, noPts, valueX, out
valueXactualSize, noPts, valueY, out valueYactualSize, noPts, valueFailedY, out
valueFailedYactualSize);
```

Python example:

```
status, dataValueX, dataIndexXactualSize, dataValueY, dataValueYactualSize,
dataValueFailedY, dataValueFailedYactualSize =
ANVNA_GetUpperLimitFailPointsBuffer(sessionId, 'CH1:TR1', 4, 4, 4)
```

MATLAB example:

```
[status, dataValueX, dataValueXActualSize, dataValueY, dataValueYActualSize,
dataValueFailedY, dataValueFailedYActualSize] =
anvna.GetUpperLimitFailPointsBuffer('CH1:Trace1', noPoints, noPoints,
```

B-151 ANVNA_GetLowerTraceLowerLimitBuffer

```
ViStatus ANVNA_GetLowerTraceLowerLimitBuffer (
  ViUInt32 Vi,
  ViConstString RepCapIdentifier,
  ViInt32 dataIndexXSize,
  ViPInt32 dataIndexX,
  ViPInt32 dataIndexXActualSize,
  ViInt32 dataValueYSize,
  ViPReal32 dataValueY,
  ViPInt32 dataValueYActualSize );
```

:

Purpose: Returns lower limit buffer for lower trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier.	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability
dataIndexXSize	ViInt32	Number of elements in indexX
dataIndexX	ViInt32	Index X values
dataIndexXActualSize	ViInt32	Actual number of elements in IndexX
dataValueYSize	ViInt32	Number of elements in dataValueY.
dataValueY	ViReal64[]	Output buffer containing values Y.
dataValueYActualSize	ViPInt32	Actual number of elements in dataValueY.

C++ example:

```
ViInt32 noOfIndexX = 10;
ViInt32 noOfValueY = 10;
ViInt32 outXActualSize;
ViInt32 *dataIndexX = new ViInt32 [10];
ViInt32 outYActualSize;
ViReal64 *dataValueY = new ViReal64 [10];

status = ANVNA_GetLowerTraceLowerLimitBuffer ( sessionId, “CH1:TR1”, noOfIndexX,
dataIndexX, &outXActualSize, noOfValueY, dataValueY, &outYActualSize);
```

C# example:

```
        int noPts = 10;
        int[] indexX = new int[noPts];
        int indexXactualSize;
  double[] valueY = new double[noPts];
        int valueYactualSize;


      ANVNA.GetLowerTraceLowerLimitBuffer (sessionId, “CH1:TR1”, noPts, indexX, out
  indexXactualSize, noPts, valueY, out valueYactualSize);
```

Python example:

```
  status, dataIndexX, dataIndexXactualSize, dataValueY, dataValueYActualSize =
  ANVNA_GetLowerTraceLowerLimitBuffer (sessionId, 'CH1:TR1', noPts, noPts)
```

MATLAB example:

[status, dataIndexX, dataIndexXActualSize, dataValueY, dataValueYActualSize] = anvna.GetLowerTraceLowerLimitBuffer ('CH1:Trace1', noPoints, noPoints);

B-152 ANVNA_GetLowerTraceUpperLimitBuffer

```
ViStatus ANVNA_GetLowerTraceUpperLimitBuffer (
  ViUInt32 Vi,
  ViConstString RepCapIdentifier,
  ViInt32 dataIndexXSize,
  ViPInt32 dataIndexX,
  ViPInt32 dataIndexXActualSize,
  ViInt32 dataValueYSize,
  ViPReal32 dataValueY,
  ViPInt32 dataValueYActualSize );
```

:

Purpose: Returns upper limit buffer for lower trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier.	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataIndexXSize	ViInt32	Number of elements in indexX
dataIndexX	ViInt32	Index X values
dataIndexXActualSize	ViInt32	Actual number of elements in IndexX
dataValueYSize	ViInt32	Number of elements in dataValueY.
dataValueY	ViReal64[]	Output buffer containing values Y.
dataValueYActualSize	ViPInt32	Actual number of elements in dataValueY

C++ example:

```
     ViInt32 noOfPts = 10;
ViInt32 outXActualSize;
ViInt32 *dataIndexX = new ViInt32 [10];
ViInt32 outYActualSize;
ViReal64 *dataValueY = new ViReal64 [10];

status = ANVNA_GetLowerTraceUpperLimitBuffer ( sessionId, “CH1:TR1”, noOfPts,
dataIndexX, &outXActualSize, noOfPts, dataValueY, &outYActualSize);
```

C# example:

```
      int noPts = 10;
      int[] indexX = new int[noPts];
      int indexXactualSize;
double[] valueY = new double[noPts];
      int valueYactualSize;


     ANVNA.GetLowerTraceUpperLimitBuffer (sessionId, “CH1:TR1”, noPts, indexX, out
indexXactualSize, noPts, valueY, out valueYactualSize);
```

Python example:

```
status, dataIndexX, dataIndexXactualSize, dataValueY, dataValueYActualSize =
ANVNA_GetLowerTraceUpperLimitBuffer (sessionId, 'CH1:TR1', 10, 10)
```

MATLAB example:

[status, dataIndexX, dataIndexXActualSize, dataValueY, dataValueYActualSize] = anvna.GetLowerTraceUpperLimitBuffer ('CH1:Trace1', noPoints, noPoints);

B-153 ANVNA_GetLowerTraceLowerLimitFailPointsBuffer

```
ViStatus ANVNA_GetLowerTraceLowerLimitFailPointsBuffer (
  ViUInt32 Vi,
  ViConstString RepCapIdentifier,
  ViInt32 dataValueXSize,
  ViPReal64 dataValueX,
  ViPInt32 dataValueXActualSize);
```

:

Purpose: Returns lower limit fail points buffer for lower trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier.	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataValueXSize	ViInt32	Number of values X
dataValuesX	ViPReal64	X values
dataValuesXActualSize	ViPInt32	Actual number of X values

C++ example:

```
     ViInt32 noOfPts = 10;
ViInt32 outXActualSize;
ViReal64 *dataValueX = new ViReal64 [10];

status = ANVNA_GetLowerTraceLowerLimitFailPointersBuffer ( sessionId, “CH1:TR1”,
noOfPts, dataValueX, &outXActualSize);
```

C# example:

```
      int noPts = 10;
double[] valueX = new double[noPts];
      int valueXactualSize;

     ANVNA.GetLowerTraceLowerLimitFailPointersBuffer (sessionId, “CH1:TR1”, noPts,
valueX, out valueXactualSize);
```

Python example:

```
status, dataValueX, dataValueXActualSize =
ANVNA_GetLowerTraceLowerLimitFailPointersBuffer (sessionId, 'CH1:TR1', 10)
```

MATLAB example:

[status, dataValueX, dataValueXActualSize] = anvna.GetLowerTraceLowerLimitFailPointersBuffer ('CH1:Trace1', noPoints);

B-154 ANVNA_GetLowerTraceUpperLimitFailPointsBuffer

```
ViStatus ANVNA_GetLowerTraceUpperLimitFailPointsBuffer (
  ViUInt32 Vi,
  ViConstString RepCapIdentifier,
  ViInt32 dataValueXSize,
  ViPReal64 dataValueX,
  ViPInt32 dataValueXActualSize);
```

:

Purpose: Returns upper limit fail points buffer for lower trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.

Name	Variable Type	Description
repCapIdentifier.	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataValueXSize	ViInt32	Number of values X
dataValuesX	ViPReal64	X values
dataValuesXActualSize	ViPInt32	Actual number of X values

C++ example:

```
      ViInt32 noOfPts = 10;
  ViInt32 outXActualSize;
  ViReal64 *dataValueX = new ViReal64 [10];

  status = ANVNA_GetLowerTraceUpperLimitFailPointersBuffer ( sessionId, “CH1:TR1”,
  noOfPts, dataValueX, &outXActualSize);
```

C# example:

```
       int noPts = 10;
  double[] valueX = new double[noPts];
       int valueXactualSize;

      ANVNA.GetLowerTraceUpperLimitFailPointersBuffer (sessionId, “CH1:TR1”, noPts,
  valueX, out valueXactualSize);
```

Python example:

```
  status, dataValueX, dataValueXActualSize =
  ANVNA_GetLowerTraceLowerUpperFailPointersBuffer (sessionId, 'CH1:TR1', 10)
```

MATLAB example:

[status, dataValueX, dataValueXActualSize] = anvna.GetLowerTraceUpperLimitFailPointersBuffer ('CH1:Trace1', noPoints);

B-155 ANVNA_SetRippleLimitTestingOnOff

```
ViStatus ANVNA_SetRippleLimitTestingOnOff (
  ViSession vi,
```

```
ViConstString repCapIdentifier,
ViBoolean onOff);
```

Purpose: Activates the ripple limit testing if the boolean parameter is set on true of deactivates it otherwise.Checks if the limit test is failing or not for a channel trace.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
onOff	ViBoolean	Ripple limit testing state.

C++ example:

```
status = ANVNA_SetRippleLimitTestingOnOff ( sessionId, “CH1:TR1”, VI_TRUE );
```

C# example:

```
ANVNA.SetRippleLimitTestingOnOff(sessionId, “CH1:TR1”, 1);
```

Python example:

```
status = ANVNA_SetRippleLimitTestingOnOff(sessionId, 'CH1:TR1', 1)
```

MATLAB example:

```
status = anvna.SetRippleLimitTestingOnOff('CH1:Trace1', 1);
```

B-156 ANVNA_GetRippleLimitTestingOnOff

```
ViStatus ANVNA_GetRippleLimitTestingOnOff (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViBoolean onOff);
```

:

Purpose: Returns the state of ripple limit testing query, declared per trace. Default value is 1.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
onOff	ViBoolean	Ripple limit testing state.

C++ example:

```
    sViBoolean rippleLimitTestingStatus = VI_FALSE;
status = ANVNA_GetRippleLimitTestingOnOff ( sessionId, “CH1:TR1”,
&rippleLimitTestingStatus );
```

C# example:

```
ushort onOff;
     ANVNA.GetRippleLimitTestingOnOff(sessionId, “CH1:TR1”, out onOff);
```

Python example:

```
status, limitSet = ANVNA_GetRippleLimitTestingOnOff(sessionId, 'CH1:TR1')
print('Limit testing get value is {0}'.format(limitSet))
```

MATLAB example:

```
[status, limitTesting] = anvna.GetRippleLimitTestingOnOff('CH1:Trace1');
```

B-157 ANVNA_SetRippleLimitTestResultSign

```
ViStatus ANVNA_SetRippleLimitTestResultSign(
  ViSession vi,
  ViBoolean onOff)
```

:

Purpose: Activates the ripple limit testing result sign if the boolean parameter is set on true of deactivates it otherwise.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
onOff	ViBoolean	Ripple limit testing sign state.

C++ example:

```
status = ANVNA_SetRippleLimitTestResultSign ( sessionId, VI_TRUE );
```

C# example:

```
ANVNA.SetRippleLimitTestResultSign(sessionId, 1);
```

Python example:

```
status = ANVNA_SetRippleLimitTestResultSign(sessionId, 1)
```

MATLAB example:

```
status = anvna.SetRippleLimitTestResultSign(1);
```

B-158 ANVNA_GetRippleLimitTestResultSign

```
ViStatus ANVNA_GetRippleLimitTestResultSign(
  ViSession vi,
  ViBoolean onOff)
```

:

Purpose: Returns the state of ripple limit testing result sign declared per trace. Default value is 0.

Parameter List:

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session
onOff	ViBoolean	Ripple limit testing result sign state.

C++ example:

```
ViBoolean rippleLimitTestingSignStatus = VI_FALSE;
status = ANVNA_GetRippleLimitTestResultSign ( sessionId,
&rippleLimitTestingSignStatus );
```

C# example:

```
ushort limitSign;
      ANVNA.GetRippleLimitTestResultSign(sessionId, out limitSign);
```

Python example:

```
status, limitVisible = ANVNA_GetRippleLimitTestResultSign(sessionId)
  print('Limit testing sign get value is {0}'.format(limitVisible))
```

MATLAB example:

```
[status, limitSignTesting] = anvna.GetRippleLimitTestResultSign();
```

B-159 ANVNA_AddDefaultRippleLimitSegment

```
ViStatus ANVNA_ AddDefaultRippleLimitSegment(
  ViStatus ANVNA_ AddDefaultRippleLimitSegment
  ViBoolean onOff)
```

:

Purpose: Add default ripple limit segment for a channel trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.

C++ example:

```
status = ANVNA_ AddDefaultRippleLimitSegment ( sessionId, “CH1:TR1” );
```

C# example:

```
ANVNA. AddDefaultRippleLimitSegment (sessionId, “CH1:TR1”);
```

Python example:

```
status = ANVNA_ AddDefaultRippleLimitSegment (sessionId, 'CH1:TR1')
```

MATLAB example:

```
status = anvna. AddDefaultRippleLimitSegment ('CH1:Trace1');
```

B-160 ANVNA_GetRippleLimitsCount

```
ViStatus ANVNA_GetRippleLimitsCount(
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt32 readCount);
```

:

Purpose: Returns the number of ripple limits declared per trace. Default value is 0. The function will return 0 if no segments are declared or after ANVNA_ClearAllRippleLimits is called.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
readCount	ViPUInt32	Number of ripple limits returned by the function

C++ example:

```
ViUInt32 limits = 0;
status = ANVNA_GetRippleLimitsCount ( sessionId, “CH1:TR1”, &limits );
printf(“Number of limits is %d\n”, limits );
```

C# example:

```
UInt32 limitsCount;
      ANVNA.GetRippleLimitsCount(sessionId, “CH1:TR1”, out limitsCount);
```

Python example:

```
status, limitsCount = ANVNA_GetRippleLimitsCount(sessionId, 'CH1:TR1')
print('Limit number is {0}'.format(limitsCount))
```

MATLAB example:

```
[status, noOfRippleLimits] = anvna.GetRippleLimitsCount('CH1:Trace1');
```

B-161 ANVNA_DeleteRippleLimitSegment

```
ViStatus ANVNA_DeleteRippleLimitSegment(
  ViSession vi,
  ViConstString repCapIdentifier);
```

:

Purpose: Deletes ripple limit segment for a channel trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
readCount	ViPUInt32	Number of ripple limits returned by the function

C++ example:

```
status = ANVNA_DeleteRippleLimitSegment( sessionId, “CH1:TR1” );
```

C# example:

```
ANVNA.DeleteRippleLimitSegment(sessionId, “CH1:TR1”);
```

Python example:

```
status = ANVNA_DeleteRippleLimitSegment(sessionId, 'CH1:TR1')
```

MATLAB example:

```
status = anvna.DeleteRippleLimitSegment('CH1:Trace1');
```

B-162 ANVNA_ClearAllRippleLimits

```
ViStatus ANVNA_ClearAllRippleLimits(
 ViSession vi,
 ViConstString repCapIdentifier);
```

:

Purpose: Deletes all the ripple limits for a channel trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.

C++ example:

```
status = ANVNA_ClearAllRippleLimits ( sessionId, “CH1:TR1” );
```

C# example:

```
ANVNA.ClearAllRippleLimits(sessionId, “CH1:TR1”);
```

Python example:

```
status = ANVNA_ClearAllRippleLimits(sessionId, 'CH1:TR1')
```

MATLAB example:

```
status = anvna.ClearAllRippleLimits('CH1:Trace1');
```

B-163 ANVNA_AddRippleLimitSegment

```
ViStatus ANVNA_AddRippleLimitSegment ((
 ViSession Vi,
 ViConstString RepCapIdentifier,
 ViReal64 x1,
 ViReal64 x2,
 ViReal32 ripple );
```

Purpose: Adds a new ripple limit segment.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
x1	ViReal64	Start frequency in Hz.
x2	ViReal64	Stop frequency in Hz.
ripple	ViReal32	ripple value

C++ example:

```
ViReal64 startFrequency = 3000000000;
ViReal64 stopFrequency = 8000000000;
ViReal32 ripple = 9.8;

status = ANVNA_AddRippleLimitSegment ( sessionId, “CH1:TR1”, startFrequency,
stopFrequency, ripple );
```

C# example:

```
ANVNA.AddRippleLimitSegment(sessionId, “CH1:TR1”, startFrequency, stopFrequency,
(float) -9.8);
```

Python example:

```
x1 = float(10000000); x2 = float(8000000000); ripple = -99.8;
status = ANVNA_AddRippleLimitSegment(sessionId, 'CH1:TR1', x1, x2, ripple)
```

MATLAB example:

```
status = anvna.AddRippleLimitSegment('CH1:Trace1', startFrequency, stopFrequency,
-9.4);
```

B-164 ANVNA_GetRippleLimitValues

```
ViStatus ANVNA_GetRippleLimitValues (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViPReal64 x1,
  ViPReal64 x2,
  ViPReal32 ripple );
```

:

Purpose: Get configuration for an existing ripple limit, Start and Stop frequency, ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
x1	ViReal64	Start frequency in Hz.
x2	ViReal64	Stop frequency in Hz.
ripple	ViReal32	ripple value

C++ example:

```
ViReal64 startFrequency = 0;
ViReal64 stopFrequency = 0;
ViReal32 ripple = 0;
status = ANVNA_GetRippleLimitValues ( sessionId, “CH1:TR1”, 1,
&startFrequency, &stopFrequency, &ripple );
```

C# example:

```
double outX1, outX2;
      float ripple;
   ANVNA.GetRippleLimitValues(sessionId, “CH1:TR1”, 1, out outX1, out outX2, out
ripple);
```

Python example:

```
    status, outX1, outX2, ripple = ANVNA_GetRippleLimitValues(sessionId, 'CH1:TR1',
1);
```

MATLAB example:

```
[status, outStartFreq, outStopFreq, outRipple] =
anvna.GetRippleLimitValues('CH1:Trace1', 1);
```

B-165 ANVNA_DeleteRippleLimitSegmentAt

```
ViStatus ANVNA_DeleteRippleLimitSegmentAt (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 limitSegmentNumber);
```

:

Purpose: Deletes ripple limit segment at index specified by limitSegmentNumber, for a trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Ripple limit index number.

C++ example:

```
status = ANVNA_DeleteRippleLimitSegmentAt ( sessionId, “CH1:TR1”, 1 );
```

C# example:

```
ANVNA.DeleteRippleLimitSegmentAt(sessionId, “CH1:TR1”, 1);
```

Python example:

```
status = ANVNA_DeleteRippleLimitSegmentAt(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
status = anvna.DeleteRippleLimitSegmentAt('CH1:Trace1', 1);
```

B-166 ANVNA_IsRippleLimitTestPass

```
ViStatus ANVNA_IsRippleLimitTestPass (
    VViSession vi,
    ViConstString repCapIdentifier,
    ViPBoolean passNoPass );
```

:

Purpose: Checks if the ripple limit test is failing or not for a channel trace.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
passNoPass	ViBoolean	RRipple limit testing result.

C++ example:

```
    ViBoolean limitTestingResultStatus = VI_FALSE;
status = ANVNA_IsRippleLimitTestPass ( sessionId,
“CH1:TR1”,&limitTestingResultStatus );
```

C# example:

```
ushort outLimitTestPass;
      ANVNA.IsRippleLimitTestPass(sessionId, “CH1:TR1”, out outLimitTestPass);
```

Python example:

```
status, isLimitTestPass = ANVNA_IsRippleLimitTestPass(sessionId, 'CH1:TR1')
```

MATLAB example:

```
[status, limitPass] = anvna.IsRippleLimitTestPass('CH1:Trace1');
```

B-167 ANVNA_SetRippleLimitX1Val

```
ViStatus ANVNA_SetRippleLimitX1Val (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViReal64 x1);
```

:

Purpose: Set configuration for an existing ripple limit using Start and Stop frequency, ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
x1	ViReal64	x1 value in Hz.

C++ example:

```
      ViUInt32 segmentNumber = 1;
ViReal64 x1 = 10000000;
status = ANVNA_SetRippleLimitX1Val ( sessionId, “CH1:TR1”, segmentNumber, x1 );
```

C# example:

```
ANVNA.SetRippleLimitX1Val(sessionId, “CH1:TR1”, (uint)1, x1);
```

Python example:

```
 x1 = float(10000000;
status = ANVNA_SetRippleLimitX1Val(sessionId, 'CH1:TR1', 1, x1)
```

MATLAB example:

```
status = anvna.SetRippleLimitX1Val('CH1:Trace1', 1, x1);
```

B-168 ANVNA_GetRippleLimitX1Val

```
ViStatus ANVNA_GetRippleLimitX1Val (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViReal64 x2);
```

:

Purpose: Get configuration for an existing ripple limit using Start and Stop frequency, ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Ripple limit index number.
x1	ViReal64	x1 value in Hz.

C++ example:

```
      ViUInt32 segmentNumber = 1;
ViReal64 x1 = 0;
status = ANVNA_GetRippleLimitX1Val ( sessionId, “CH1:TR1”, segmentNumber, &x1 );
```

C# example:

```
double outX1;
ANVNA.GetRippleLimitX1Val(sessionId, “CH1:TR1”, 1, out outX1);
```

Python example:

```
status, outX1= ANVNA_GetRippleLimitX1Val(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
[status, outX1] = anvna.GetRippleLimitX1Val('CH1:Trace1', 1);
```

B-169 ANVNA_SetRippleLimitX2Val

```
ViStatus ANVNA_SetRippleLimitX2Val(
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViReal64 x2);
```

:

Purpose: Set configuration for an existing ripple limit using Start and Stop frequency, ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
x2	ViReal64	x2 value in Hz.

C++ example:

```
      ViUInt32 segmentNumber = 1;
ViReal64 x2 = 10000000;
status = ANVNA_SetRippleLimitX2Val ( sessionId, “CH1:TR1”, segmentNumber, x2 );
```

C# example:

```
double outX1;
ANVNA.GetRippleLimitX1Val(sessionId, “CH1:TR1”, 1, out outX1);
```

Python example:

```
X2 = float(10000000;
status = ANVNA_SetRippleLimitX2Val(sessionId, 'CH1:TR1', 1, x2)
```

MATLAB example:

```
status = anvna.SetRippleLimitX2Val('CH1:Trace1', 1, x2);
```

B-170 ANVNA_GetRippleLimitX2Val

```
ViStatus ANVNA_GetRippleLimitX2Val (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViPReal64 x2);
```

:

Purpose: Get configuration for an existing ripple limit, Start and Stop frequency, ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Ripple limit index number.
x2	ViReal64	x2 value in Hz.

C++ example:

```
      ViUInt32 segmentNumber = 1;
ViReal64 x2 = 0;
status = ANVNA_GetRippleLimitX2Val ( sessionId, “CH1:TR1”, segmentNumber, &x2 );
```

C# example:

```
double outX2;
ANVNA.GetRippleLimitX1Val(sessionId, “CH1:TR1”, 1, out outX2);
```

Python example:

```
status, outX2= ANVNA_GetRippleLimitX2Val(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
[status, outX2] = anvna.GetRippleLimitX2Val('CH1:Trace1', 1);
```

B-171 ANVNA_SetRippleLimitRippleVal

```
ViStatus ANVNA_SetRippleLimitRippleVal (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViReal32 ripple);
```

:

Purpose: Set configuration for an existing ripple limit using Start and Stop frequency, ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Limit index number.
ripple	ViReal32	ripple value.

C++ example:

```
      ViUInt32 segmentNumber = 1;
ViReal32 ripple = 10;
status = ANVNA_SetRippleLimitRippleVal ( sessionId, “CH1:TR1”, segmentNumber,
ripple );
```

C# example:

```
ANVNA.SetRippleLimitRippleVal(sessionId, “CH1:TR1”, (uint)1, ripple);
```

Python example:

```
ripple = float(10);
status = ANVNA_SetRippleLimitRippleVal(sessionId, 'CH1:TR1', 1, ripple)
```

MATLAB example:

```
status = anvna.SetRippleLimitRippleVal('CH1:Trace1', 1, ripple);
```

B-172 ANVNA_GetRippleLimitRippleVal

```
ViStatus ANVNA_GetRippleLimitRippleVal (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViPReal32 ripple);
```

:

Purpose: Get configuration for an existing ripple values.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViUInt32	Ripple limit index number.
ripple	ViReal32	ripple value.

C++ example:

```
      ViUInt32 segmentNumber = 1;
ViReal32 ripple = 10;
status = ANVNA_GetRippleLimitRippleVal ( sessionId, “CH1:TR1”, segmentNumber,
&ripple );
```

C# example:

```
double outRipple;
ANVNA.GetRippleLimitRippleVal(sessionId, “CH1:TR1”, 1, out outRipple);
```

Python example:

```
status, outRipple= ANVNA_GetRippleLimitRippleVal(sessionId, 'CH1:TR1', 1);
```

MATLAB example:

```
[status, outRipple] = anvna.GetRippleLimitRippleVal('CH1:Trace1', 1);
```

B-173 ANVNA_GetRippleLimitUpperLowerValues

```
ViStatus ANVNA_ GetRippleLimitUpperLowerValues (
  ViSession Vi,
  ViConstString RepCapIdentifier,
  ViUInt32 limitSegmentNumber,
  ViPUInt32 upper,
  ViPUInt32 lower);
```

:

Purpose: Get ripple limit using upper, lower parameter.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
limitSegmentNumber	ViPReal32	Limit index number.
upper	ViReal32	Output paramerer for upper.
lower	ViReal32	Output paramerer for lower.

C++ example:

```
ViPReal lower = 0;
status = ANVNA_ GetRippleLimitUpperLowerValues ( sessionId, “CH1:TR1”, 1, &upper,
&lower);
```

C# example:

```
float outUpper; float outLower;
ANVNA. GetRippleLimitUpperLowerValues (sessionId, “CH1:TR1”, 1, out outUpper, out
&outLower);
```

Python example:

```
status, outUpper outLower = ANVNA_ GetRippleLimitUpperLowerValues (sessionId,
'CH1:TR1', 1, 2);
```

MATLAB example:

```
status, outUpper, outLower = anvna. GetRippleLimitUpperLowerValues ('CH1:Trace1', 1, 2);
```

B-174 ANVNA_GetRippleLimitFailPointsBuffer

```
ViStatus ANVNA_ GetRippleLimitFailPointsBuffer (
  ViUInt32 dataSizeUpper,
  ViPUInt32 readCountUpper,
  ViPReal64 dataLower,
  ViUInt32 dataSizeLower,
  ViPUInt32 readCountLower );
```

:

Purpose: Returns lower and upper values for fails points.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier. For example “CH1:Measurement1”.
dataUpper	ViPReal64[]	Number of upper fail points.
dataSizeUpper	ViUInt32	Output paramerer for upper.
readCountUpper	ViPUInt32	Actual number of fail points.
dataLower	ViPReal64[]	Output buffer of lower fail points.
dataSizeLower	ViUInt32	Number of lower fail points.
readCountLower	ViPUInt32	Actual number of lower fail points.

C++ example:

```
ViUInt32 dataSizeUpper = 10;
ViUInt32 dataSizeLower = 15;
ViUInt32 readCountUpper;
ViUInt32 readCountLower;
ViReal64 *dataUpper = new Vireal64 [dataSizeUpper];
ViReal64 *dataLower = new ViReal64 [dataSizeLower];
status = ANVNA_GetRippleLimitFailPointsBuffer ( sessionId, “CH1:TR1”, dataUpper,
dataSizeUpper, &readCountUpper, dataLower,  dataSizeLower, &readCountLower );
delete [] dataUpper;
delete [] dataLower;
```

C# example:

```
int dataSizeUpper =10;
int dataSizeLower =15;
double[] dataUpper = new double[dataSizeUpper];
```

```
        double[] dataLower = new double[dataSizeLower];
        int readCountUpper, readCountLower;

        ANVNA.GetRippleLimitFailPointsBuffer(sessionId, “CH1:TR1”, dataUpper,
dataSizeupper, out readDataUpper, dataLower, dataSizeLower, out readDataLower);
```

Python example:

```
status, dataUpper, readCountUpper, dataLower, readCountLower = ANVNA_
GetRippleLimitFailPointsBuffer (sessionId, 'CH1:TR1', 10, 15)
```

MATLAB example:

```
[status, dataUpper, readCountUpper, dataLower, readCountLower] = anvna.
GetRippleLimitFailPointsBuffer ('CH1:Trace1', 10, 15);
```

B-175 ANVNA_GetRippleLimitLineActive

```
ViStatus ANVNA_ GetRippleLimitLineActive (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 segmentNum,
  ViPBoolean onOff);
```

:

Purpose: This function reads the activation state for the ripple limit line number.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability identifier. Ex: “CH1:TR1”.
segmentNum	ViUInt32	Segment number index starting from 1.
onOff	ViBoolean	Ripple limit’s state.

C++ example:

```
ViBoolean rippleLimitStatus = VI_FALSE;
status = ANVNA_GetRippleLimitLineActive ( sessionId, “CH1:TR1”, 1,
&rippleLimitStatus );
```

C# example:

```
ushort onOff;
      ANVNA.GetRippleLimitLineActive (sessionId, “CH1:TR1”, 1, out onOff);
```

Python example:

```
status, limitSet = ANVNA_GetRippleLimitLineActive(sessionId, 1, 'CH1:TR1')
    print('Limit testing get value is {0}'.format(limitSet))
```

MATLAB example:

```
[status, limitStatus] = anvna.GetRippleLimitLineActive ('CH1:Trace1', 1);
```

B-176 ANVNA_SetRippleLimitLineActive

```
ViStatus ANVNA_ SetRippleLimitLineActive (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 segmentNum,
  ViBoolean onOff);
```

:

Purpose: This function activates or deactivates a ripple limit line.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
segmentNum	ViUInt32	Segment number
onOff	ViBoolean	Ripple limit state.

C++ example:

```
status = ANVNA_SetRippleLimitLineActive ( sessionId, “CH1:TR1”, 1, VI_TRUE );
```

C# example:

```
ANVNA.SetRippleLimitLineActive(sessionId, “CH1:TR1”, 2, 1);
```

Python example:

```
status = ANVNA_SetRippleLimitLineActive(sessionId, 'CH1:TR1', 2, 1)
```

MATLAB example:

```
status = anvna.SetRippleLimitLineActive('CH1:Trace1', 2, 1);
```

B-177 ANVNA_SetRippleLimitLinesOnOff

```
ViStatus ANVNA_ SetRippleLimitLineActive (
  ViSession vi,
  ViConstString repCapIdentifier,
      :
```

Purpose: This function activates or deactivates ripple ripple limit lines.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
onOff	ViBoolean	Ripple limit lines state.

C++ example:

```
status = ANVNA_SetRippleLimitLinesOnOff ( sessionId, “CH1:TR1”, VI_TRUE );
```

C# example:

```
ANVNA.SetRippleLimitLinesOnOff(sessionId, “CH1:TR1”, 1);
```

Python example:

```
status = ANVNA_SetRippleLimitLinesOnOff(sessionId, 'CH1:TR1', 1)
```

MATLAB example:

```
status = anvna.SetRippleLimitLinesOnOff('CH1:Trace1', 1);
```

B-178 ANVNA_GetRippleLimitLinesOnOff

```
VViStatus ANVNA_ GetRippleLimitLinesOnOff ((
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPBoolean onOff);
```

:

Purpose: This function reads the activation state for the ripple limit lines.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is identifier. Ex: “CH1:TR1”.	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability
onOff	ViBoolean	Ripple limit’s state.

C++ example:

```
ViBoolean rippleLimitStatus = VI_FALSE;
status = ANVNA_GetRippleLimitLinesOnOff ( sessionId, “CH1:TR1”, &rippleLimitStatus
);
```

C# example:

```
ushort onOff;
      ANVNA.GetRippleLimitLinesOnOff (sessionId, “CH1:TR1”, out onOff);
```

Python example:

```
status, limitSet = ANVNA_GetRippleLimitLinesOnOff(sessionId, 'CH1:TR1')
    print('Limit testing get value is {0}'.format(limitSet))
```

MATLAB example:

```
[status, limitStatus] = anvna.GetRippleLimitLinesOnOff ('CH1:Trace1');
```

B-179 ANVNA_SetRippleLimitRippleValueFormat

```
ViStatus ANVNA_ SetRippleLimit RippleValueFormat (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 type);
```

:

Purpose: This function ripple limit value format.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier (Only channel index is parsed, ex. “CH1:TR1”)	ViConstString	The physical or virtual repeated capability identifier.
type	VUInt32	Ripple value format type.

C++ example:

```
status = ANVNA_SetRippleLimitRippleValueFormat ( sessionId, “CH1:TR1”, 1);
```

C# example:

```
ANVNA.SetRippleLimitRippleValueFormat(sessionId, “CH1:TR1”, 1);
```

Python example:

```
status = ANVNA_SetRippleLimitRippleValueFormat(sessionId, 'CH1:TR1', 1)
```

MATLAB example:

```
status = anvna.SetRippleLimitRippleValueFormat('CH1:Trace1', 1);
```

B-180 ANVNA_GetRippleLimitRippleValueFormat

```
ViStatus ANVNA_ GetRippleLimitRippleValueFormat (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViPUInt32 type);
```

:

Purpose: This function reads the ripple limit value format.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier Ex: “CH1:TR1”.	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability
type	VUInt32	Ripple limit value format.

C++ example:

```
ViUInt32 rippleLimitType = 0;
status = ANVNA_GetRippleLimitRippleValueFormat ( sessionId, “CH1:TR1”,
&rippleLimitType );
```

C# example:

```
uint type;
      ANVNA.GetRippleLimitRippleValueFormat (sessionId, “CH1:TR1”, out type);
```

Python example:

```
status, type = ANVNA_GetRippleLimitRippleValueFormat(sessionId, 'CH1:TR1')
    print('Limit testing get value is {0}'.format(type))
```

MATLAB example:

```
[status, type] = anvna.GetRippleLimitRippleValueFormat ('CH1:Trace1');
```

B-181 ANVNA_GetRippleLimitRippleMeasurementValue

```
ViStatus ANVNA_ GetRippleLimitRippleMeasurementValue (
  ViSession vi,
  ViConstString repCapIdentifier,
  ViUInt32 segmentNum,
  ViPReal64 data);
```

:

Purpose: This function get ripple limit measurement value.

Parameter List

Name	Variable Type	Description
vi	ViSession	The ViSession handle that you obtain from the ANVNA_init or ANVNA_InitWithOptions function. The handle identifies a particular instrument session.
repCapIdentifier Ex: “CH1:TR1”.	ViConstString	If the attribute applies to a repeated capability, the user passes a physical or virtual repeated capability
segmentNum	ViUInt32	Segment number index starting from 1.
data	ViPReal64	Ripple limit measurement value.

C++ example:

```
ViReal64 rippleLimitValue = 100000000;
status = ANVNA_GetRippleLimitRippleMeasurementValue ( sessionId, “CH1:TR1”, 1,
&rippleLimitValue );
```

C# example:

```
double data
ANVNA.GetRippleLimitRippleMeasurementValue (sessionId, “CH1:TR1”, 1, out data);
```

Python example:

```
status, limitValue = ANVNA_GetRippleLimitRippleMeasurementValue (sessionId, 1,
'CH1:TR1')
print('Limit testing get value is {0}'.format(limitValue))
```

MATLAB example:

```
[status, limitValue] = anvna.GetRippleLimitRippleMeasurementValue ('CH1:Trace1', 1);
```

B-182 Attributes

Attribute Information for the Following Functions

ANVNA_GetAttributeViInt32
ANVNA_SetAttributeViInt32
ANVNA_GetAttributeViInt64
ANVNA_SetAttributeViInt64
ANVNA_GetAttributeViReal64
ANVNA_SetAttributeViReal64
ANVNA_GetAttributeViSession
ANVNA_SetAttributeViSession
ANVNA_GetAttributeViBoolean
ANVNA_SetAttributeViBoolean
ANVNA_GetAttributeViString
ANVNA_SetAttributeViString

Inherent IVI Attributes

Driver Capabilities
Class Group Capabilities

ANVNA_ATTR_GROUP_CAPABILITIES

Supported Instrument Models	ANVNA_ATTR_SUPPORTED_INSTRUMENT_MODELS
Driver Identification	
Specific Driver Class Spec Major Version	ANVNA_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION
Specific Driver Class Spec Minor Version	ANVNA_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION
Specific Driver Description	ANVNA_ATTR_SPECIFIC_DRIVER_DESCRIPTION
Specific Driver Prefix	ANVNA_ATTR_SPECIFIC_DRIVER_PREFIX
Specific Driver Revision	ANVNA_ATTR_SPECIFIC_DRIVER_REVISION
Specific Driver Vendor	ANVNA_ATTR_SPECIFIC_DRIVER_VENDOR
Instrument Identification	
Instrument Firmware Revision	ANVNA_ATTR_INSTRUMENT_FIRMWARE_REVISION
Instrument Manufacturer	ANVNA_ATTR_INSTRUMENT_MANUFACTURER
Instrument Model	ANVNA_ATTR_INSTRUMENT_MODEL
User Options	
Simulate	
ANVNA_ATTR_SIMULATE	
Channel	
Averaging	ANVNA_ATTR_CHANNEL_AVERAGING

Averaging Factor	ANVNA_ATTR_CHANNEL_AVERAGING_FACTOR
Channel Count	ANVNA_ATTR_CHANNEL_COUNT
Correction	ANVNA_ATTR_CHANNEL_CORRECTION
CW Frequency	ANVNA_ATTR_CHANNEL_CW_FREQUENCY
IF Bandwidth	ANVNA_ATTR_CHANNEL_IF_BANDWIDTH
Number	ANVNA_ATTR_CHANNEL_NUMBER
Points	ANVNA_ATTR_CHANNEL_POINTS
Sweep Type	ANVNA_ATTR_CHANNEL_SWEEP_TYPE
Trigger Mode	ANVNA_ATTR_CHANNEL_TRIGGER_MODE
Measurement	
Channel Measurement Count	ANVNA_ATTR_CHANNEL_MEASUREMENT_COUNT
Format	ANVNA_ATTR_CHANNEL_MEASUREMENT_FORMAT
Smoothing	ANVNA_ATTR_CHANNEL_MEASUREMENT_SMOOTHING
Smoothing Aperture	
ANVNA_ATTR_CHANNEL_MEASUREMENT_SMOOTHING_APERTURE	
StimulusRange	
Center	ANVNA_ATTR_CHANNEL_STIMULUSRANGE_CENTER
Span	ANVNA_ATTR_CHANNEL_STIMULUSRANGE_SPAN
Start	ANVNA_ATTR_CHANNEL_STIMULUSRANGE_START
Stop	ANVNA_ATTR_CHANNEL_STIMULUSRANGE_STOP
System	
Serial Number	ANVNA_ATTR_SYSTEM_SERIAL_NUMBER
Trigger	
Source	ANVNA_ATTR_TRIGGER_SOURCE

ANVNA_ATTR_CHANNEL_AVERAGING

Data Type: ViBoolean

Description: Turns trace averaging ON or OFF.

True or 1 = ON

False or 0 = OFF

Read/write attribute, accessible via ANVNA_GetAttributeViBoolean and ANVNA_SetAttributeViBoolean

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

ANVNA_ATTR_CHANNEL_AVERAGING_FACTOR

Data Type:

ViUInt32

Description: Sets the number of measurement sweeps to combine for an average.

Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

WAITING FOR VALID RANGE

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

ANVNA_ATTR_CHANNEL_CORRECTION

Data Type

ViBoolean

Description: Sets the correction state (calibration state) for all measurements on the channel.

Read/write attribute accessible via ANVNA_GetAttributeViBoolean and ANVNA_SetAttributeViBoolean functions.

True/1=ON

False/0=OFF

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

ANVNA_ATTR_CHANNEL_COUNT

Data Type: ViUInt32

Accepted Values: 1, 2, 3, 4, 6, 8, 9, 10, 12, 16

repCapIdentifier info for Set/Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Attribute that contains the number of Channels available on the instrument. This is the maximum index that may be used with the GetChannelName() function. Maximum value is 16. Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

C++ example:

```
/*
 * Initialization code ...
 */
// Create two measurement channels:
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
if ((status = ANVNA_SetAttributeViInt32(session, "", ANVNA_ATTR_CHANNEL_COUNT, 2)) != VI_SUCCESS)
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while creating channels: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
}
```

ANVNA_ATTR_CHANNEL_CW_FREQUENCY

Data Type

ViReal64

Description: Sets the Continuous Wave frequency.

Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions.

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

To E ADDED: VALID RANGE OF VALUES

MEASUREMENT UNIT: Hz

ANVNA_ATTR_CHANNEL_IF_BANDWIDTH

Data Type

ViReal64

Description: Sets the bandwidth of the digital IF filter to be used in the measurement.

MODEL DEPENDENT . TO BE ADDED - ACCEPTED VALUES FOR ALL MODELS

VALUES IN HZ

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions

ANVNA_ATTR_CHANNEL_MEASUREMENT_COUNT

Data Type

ViUInt32

Description: Attribute that contains the number of Measurements available on the instrument for one channel. This is the maximum index that may be used with the GetChannelMeasurementName() function.

Maximum value is 16.

RANGE: 1-16

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

C++ Example:

```
/*
 * Initialization code ...
 */
//Set two traces for channel 1:
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
if ((status = ANVNA_SetAttributeViUInt32(session, "CH1:", ANVNA_ATTR_CHANNEL_MEASURE-
MENT_COUNT, 2)) != VI_SUCCESS)
```

```
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while creating traces for channel 1: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
}
```

ANVNA_ATTR_CHANNEL_MEASUREMENT_FORMAT

Data Type: ViUInt32

Description: Sets the data format for the specified measurement

Values Definition:

#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_LOG_MAG	0
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_LIN_MAG	1
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_PHAS	2
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_GROUP_DELAY	3
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_SWR	4
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_REAL	5
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_IMAG	6
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_POLAR	7
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_SMITH	8
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_S_LINEAR	9
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_S_LOGARITHMIC	10
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_S_COMPLEX	11
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_S_ADMITTANCE	12
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_P_LINEAR	13
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_P_LOGARITHMIC	14
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_U_PHASE	15
#define ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_P_PHASE	16

Accepted Values: 0, 1, 2, 3, 4, 5, 6, 8

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

C++ example:

```
        /*
         * Initialization code ...
         */
        //Set trace format to logarithmic magnitude for trace 1 in channel 1:
        ViStatus status = VI_SUCCESS;
        char ErrorMessage[MAX_STRING_LENGTH];
        if ((status = ANVNA_SetAttributeViUInt32(sessionId, "CH1:Trace1", ANVNA_ATTR_CHANNEL_MEASURE-
MENT_FORMAT, ANVNA_VAL_ANRITSU_VNA_MEASUREMENT_LOG_MAG)) != VI_SUCCESS)
        {
                ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
                printf("Error %d while setting trace format on channel 1, trace 1: %s\n", status, ErrorMessage);
                ANVNA_close(session);
                exit(1);
        }
```

ANVNA_ATTR_CHANNEL_MEASUREMENT_SMOOTHING

Data Type

ViBoolean

Description: Sets smoothing on or off for a measurement (trace)

Read/write attribute accessible via ANVNA_GetAttributeViBoolean and ANVNA_SetAttributeViBoolean functions.

True/1=ON

False/0=OFF

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

ANVNA_ATTR_CHANNEL_MEASUREMENT_SMOOTHING_APERTURE

Data Type

ViReal64

Description: Sets the value of smoothing percentage for the measurement

Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions.

RANGE 0-100(percent) – NO MEASUREMENT UNIT

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

ANVNA_ATTR_CHANNEL_POINTS

Data Type

ViUInt32

Description: Sets the number of data points for the measurement, for the specified channel.

Maximum value depends on the model as follows:

//TODO: define the maximum number of points per model

Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

RANGE IS MODEL DEPENDENT

MS46121A/B: 0-20001

MS46122A/B: 0-16001

MS46322A/B: 0-16001

MS4652xB: 0-20001

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

C++ example:

```
/*
 * Initialization code ...
 */
// set number of points to 201 on channel 1:
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
if ((status = ANVNA_SetAttributeViUInt32(sessionId, "CH1:", ANVNA_ATTR_CHANNEL_POINTS, 201)) != VI_SUCCESS)
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while setting trace format on channel 1, trace 2: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
}
```

ANVNA_ATTR_CHANNEL_STIMULUSRANGE_CENTER

Data Type

ViReal64

Description: Sets the center frequency value of the sweep for the channel

Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions.

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

MEASUREMENT UNIT: HZ

ANVNA_ATTR_CHANNEL_STIMULUSRANGE_SPAN

Data Type
ViReal64
Description: Sets the span value of the sweep for the channel
Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions.
repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

MEASUREMENT UNIT: HZ

ANVNA_ATTR_CHANNEL_STIMULUSRANGE_START

repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

Data Type
ViReal64
Description: Sets the start value of the sweep range of channel
Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions.
MEASUREMENT UNIT: HZ

C++ example:

```
/*
 * Initialization code ...
 */
// set start frequency to 10 MHz on channel 1:
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
if ((status = ANVNA_SetAttributeViReal64(sessionId, "CH1:", ANVNA_ATTR_CHANNEL_STIMULUS-
RANGE_START, 10000000)) != VI_SUCCESS)
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while setting start frequency on channel 1: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
}
```

ANVNA_ATTR_CHANNEL_STIMULUSRANGE_STOP

Data Type

ViReal64

Description: Sets the stop value of the sweep range of channel

Read/write attribute accessible via ANVNA_GetAttributeViReal64 and ANVNA_SetAttributeViReal64 functions. repCapIdentifier info for Set/Get Attribute functions: Channel and Trace/Measurement names expected to be passed in string, eg. "CH1:Measurement1"

MEASUREMENT UNIT: HZ

C++ Example:

```
/*
 * Initialization code ...
 */
// set stop frequency to 8 GHz on channel 1:
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
if ((status = ANVNA_SetAttributeViReal64(sessionId, "CH1:", ANVNA_ATTR_CHANNEL_STIMULUS-
RANGE_STOP, 8000000000)) != VI_SUCCESS)
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while setting stop frequency on channel 1: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
}
```

STOP VALUES ARE MODEL DEPENDENT

ANVNA_ATTR_CHANNEL_STIMULUSRANGE_START VALUE is 300 KHz

ANVNA_ATTR_CHANNEL_STIMULUSRANGE_STOP values:

- MS46524B and MS46522B: 8.5 GHz
- MS322A DEPENDING ON MODEL, UP TO 4/8/14/20/30/43.5 GHz
- MS122A DEPENDING ON MODEL, UP TO 8/20/43.5 GHz
- MS121A DEPENDING ON MODEL, UP TO 4/6 GHz

ANVNA_ATTR_CHANNEL_SWEEP_TYPE

Data Type

ViUInt32

Description: Sets the sweep type of channel.

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

Possible values are:

ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_LIN_FREQUENCYlinear frequency sweep

ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_SEGMENTsegmented sweep

ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_INDEX_SEGMENTsegmented sweep index based (?)

ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_POWERpower sweep

Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

VALUES DEFINITIONS:

Definitions	Values
#define ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_LIN_FREQUENCY	**0**
#define ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_LOG_FREQUENCY	**1**
#define ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_SEGMENT	**2**
#define ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_POWER	**3**
#define ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_CW_TIME	**4**
#define ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_INDEX_SEGMENT	**5**

Currently accepted values are 0, 2, 3, and 5.

C++ example:

```
/*
 * Initialization code ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
// linear continuous sweep (not segmented)
if ((status = ANVNA_SetAttributeViUInt32(sessionId, channel, ANVNA_ATTR_CHANNEL_SWEEP_TYPE,
ANVNA_VAL_ANRITSU_VNA_SWEEP_TYPE_LIN_FREQUENCY)) != VI_SUCCESS)
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while setting linear continuous sweep on channel 1: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
```

```
}
```

ANVNA_ATTR_CHANNEL_TRIGGER_MODE

Data Type

ViUInt32

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

Description: Sets the trigger mode for the specified channel.

Possible values are:

0 ANVNA_VAL_TRIGGER_MODE_HOLD

1 ANVNA_VAL_TRIGGER_MODE_CONTINUOUS

Read/write attribute accessible via ANVNA_GetAttributeViUInt32 and ANVNA_SetAttributeViUInt32 functions.

C++ example:

```
/*
 * Initialization code ...
 */
ViStatus status = VI_SUCCESS;
char ErrorMessage[MAX_STRING_LENGTH];
// set hold state to true on channel 1 - hold after every sweep
if ((status = ANVNA_SetAttributeViInt32(sessionId, "CH1:", ANVNA_ATTR_CHANNEL_TRIGGER_MODE,
ANVNA_VAL_TRIGGER_MODE_HOLD)) != VI_SUCCESS)
{
        ANVNA_error_message(session, status, ErrorMessage, MAX_STRING_LENGTH);
        printf("Error %d while setting trigger mode to hold on channel 1: %s\n", status, ErrorMessage);
        ANVNA_close(session);
        exit(1);
}
```

ANVNA_ATTR_GROUP_CAPABILITIES

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: A comma-separated list of the class capability groups implemented by the driver. Capability group names are documented in the IVI class specifications. If the driver is not class compliant, the driver returns an empty string.

Read only attribute, ANVNA_GetAttributeViString returns empty string.

ANVNA_ATTR_INSTRUMENT_FIRMWARE_REVISION

Data Type
ViString
repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: The firmware revision reported by the physical instrument (FPGA version).
Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repcap parameter.
MODEL-DEPENDENT STRING

ANVNA_ATTR_INSTRUMENT_MANUFACTURER

Data Type
ViString
repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: The name of the manufacturer reported by the physical instrument ("Anritsu").
Manufacturer is limited to 256 bytes
Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repcap parameter.
EXPECTED VALUE=”Anritsu”

ANVNA_ATTR_INSTRUMENT_MODEL

Data Type
ViString
repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: The model number or name reported by the physical instrument.
Model is limited to 256 bytes
Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repcap parameter.
EXPECTED VALUES : ”MS46322A/B”, “MS46522B”, “MS46524B”

ANVNA_ATTR_SIMULATE

Data Type
ViBoolean
repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: If True, the driver does not perform I/O to the instrument, andreturns simulated values for output parameters.
Read only attribute accessible via ANVNA_GetAttributeViBoolean, not dependent on the repcap parameter.
True or 1 = ON
False or 0 = OFF

ANVNA_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MAJOR_VERSION

Data Type

ViInt32

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Returns the major version number of the class specification in accordance with which the IVI specific driver was developed. Zero is returned if the driver is not compliant with a class specification.

Read only attribute accessible via ANVNA_GetAttributeViInt32, not dependent on the repeated capability parameter.

EXPECTED VALUE='1' (CURRENTLY)

ANVNA_ATTR_SPECIFIC_DRIVER_CLASS_SPEC_MINOR_VERSION

Data Type

ViInt32

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Returns the minor version number of the class specification in accordance with which the IVI specific driver was developed. Zero is returned if the driver is not compliant with a class specification.

Read only attribute accessible via ANVNA_GetAttributeViInt32, not dependent on the repeated capability parameter.

EXPECTED VALUE='0' (CURRENTLY)

ANVNA_ATTR_SPECIFIC_DRIVER_DESCRIPTION

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Returns a brief description of the IVI specific driver. The string that this attribute returns contains a maximum of 256 bytes including the NULL byte.

Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repeated capability parameter (can be NULL).

EXPECTED VALUE=' Anritsu VNA ShockLine Driver'

ANVNA_ATTR_SPECIFIC_DRIVER_PREFIX

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Returns the case-sensitive prefix of the user-callable functions that the IVI-C specific driver exports. The string that this attribute returns contains a maximum of 32 bytes including the NULL byte.

Typically "ANVNA"

Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repeated capability parameter (can be NULL).

EXPECTED VALUE='ANVNA'

ANVNA_ATTR_SPECIFIC_DRIVER_REVISION

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Returns version information about the IVI specific driver. The string that this attribute returns contains a maximum of 256 bytes including the NULL byte.

Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repeated capability parameter (can be NULL).

EXPECTED VALUE='01'

ANVNA_ATTR_SPECIFIC_DRIVER_VENDOR

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Returns the name of the vendor that supplies the IVI specific driver. The string that this attribute returns contains a maximum of 256 bytes including the NUL byte.

Typically "Anritsu"

Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repeated capability parameter (can be NULL).

EXPECTED VALUE='Anritsu'

ANVNA_ATTR_SUPPORTED_INSTRUMENT_MODELS

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: A comma-separated list of instrument models that the IVI specific driver can control. The string does not include an abbreviation for the manufacturer if it is the same for all models.

Read only attribute accessible via ANVNA_GetAttributeViString, not dependent on the repeated capability parameter (can be NULL).

EXPECTED VALUES:' MS46322A/B','MS46522B','MS46524B' DEPENDING ON MODEL

ANVNA_ATTR_SYSTEM_SERIAL_NUMBER

Data Type

ViString

repCapIdentifier info for Get Attribute functions: No info expected to be passed in string, eg. ":" or ""

Description: Instrument serial number

Read only attribute accessible via ANVNA_GetAttributeViString function not dependent on the repeated capability parameter (can be NULL).

UNIT DEPENDENT VALUE

ANVNA_ATTR_TRIGGER_SOURCE

Data Type

ViUInt32

Description: Selects the trigger source

repCapIdentifier info for Set/Get Attribute functions: Channel name expected to be passed in string, eg. "CH1:"

Read/write attribute

Values definitions:

Definitions	**Values**
#define ANVNA_VAL_ANRITSU_VNA_TRIGGER_SOURCE_INTERNAL	0
#define ANVNA_VAL_ANRITSU_VNA_TRIGGER_SOURCE_EXTERNAL	1
#define ANVNA_VAL_ANRITSU_VNA_TRIGGER_SOURCE_BUS	2
#define ANVNA_VAL_ANRITSU_VNA_TRIGGER_SOURCE_MANUAL	3

Possible values:

0 ANVNA_VAL_ANRITSU_VNA_TRIGGER_SOURCE_INTERNAL

B-183 IVI Driver Installation

The Anritsu IVI-C driver for ShockLine VNA series can be downloaded from the Anritsu wesbsite.

There are two versions of the installation files:

- *ANVNA32-bitsWindows.exe* for 32 bit architecture, all Windows flavors;
- *ANVNA64-bitsWindows.exe* for 64 bit architecture, all Windows flavors.

As a prerequisite, the IVI foundation shared components must be installed.

See http://www.ivifoundation.org/shared_components/Default.aspx.

The ANVNA driver will use the same installation path.

C and C++

ANVNA.h and ANVNA.dll files are delivered in the standard paths as specified by IVI Foundation: <IVI Root Dir>\Include and <IVI Root Dir>\Bin respectively.

The ANVNA.lib file to be used for C/C++ application development is delivered as specified by IVI Foundation in <IVI Root Dir>\Lib\msc. There is a VS2005 C++ project example delivered in <IVI Root Dir>\Drivers\ANVNA\CPP\Examples.

C#

C Sharp binaries ANVNA_IVI_CSHARP.dll and ANVNA_IVI_CSHARP_LIB.dll are also delivered in <IVI Root Dir>\Bin because this path is added to PATH environment variable so the dll files are available at run-time. A C Sharp VS2005 project example is delivered under <IVI Root Dir>\Drivers\ANVNA\CSHARP\Examples.

Python

For Python scripting, python version 3.4 or greater must be used. Python binaries are delivered in <IVI Root Dir>\Drivers\ANVNA\Python and a Python example can be found in <IVI Root Dir>\Drivers\ANVNA\Python\Examples.

MATLAB

MATLAB support is also provided, the ".m" driver file is delivered under <IVI Root Dir>\Drivers\ANVNA\MATLAB and an example of usage under <IVI Root Dir>\Drivers\ANVNA\MATLAB\Examples. The example has been tested with MATLAB version R2014A.

LabView

A LabView project for importing functions from the ANVNA.dll file is delivered under <IVI Root Dir>\rivers\ANVNA\LabView\ANVNA_LabView\ANVNA.lvlib file. Two examples (for calibrated and uncalibrated scenarios) are provided in <IVI Root Dir>\Drivers\ANVNA\LabView\ANVNA_LabView\EXAMPLES, tested with LabView version 13.0f2.

Note	To be able to use ShockLine devices remotely, the ShockLine application must be closed and "IVI ShockLine Service" service must be running on device.

Appendix C — Alphabetical SCPI Command List

C-1 Introduction

This appendix provides an alphabetical listing of all commands and queries available for the ShockLine Series VNA.

Sorting

Due to ASCII sorting:

- Commands beginning with a colon (“ : ” or SCPI commands) are sorted first.
- Commands with an optional keyword indicated by brackets (“[ ]”) sort after the non-optional keywords. For example :STATe sorts before [:STATe].
- Commands that start with an asterisk (“ * ” or IEEE 488.2 commands) are sorted next.
- Commands that start with a number or letter, such as IVI commands which start with letters, are sorted last.

System Suffix

The “System” suffix has been added to the ShockLine Series VNA system, diagnostic, and troubleshooting commands in this index list. These are only for identification and not a part of the actual command syntax.

C-2 Alphabetical Command List

:CALCulate{1-16}:ALL:ALTernate:TRACe:NAMe <char> 5-4
:CALCulate{1-16}:ALTernate:TRACe:NAMe:STATe <char> 5-4
:CALCulate{1-16}:ALTernate:TRACe:NAMe:STATe? 5-4
:CALCulate{1-16}:CORRection:ADAPter:REMoval 5-5
:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:X <string> 5-5
:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:X? 5-5
:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:Y <string> 5-6
:CALCulate{1-16}:CORRection:ADAPter:REMoval:CALibration:Y? 5-6
:CALCulate{1-16}:CORRection:ADAPter:REMoval:LENGth <NRf> 5-6
:CALCulate{1-16}:CORRection:ADAPter:REMoval:LENGth? 5-6
:CALCulate{1-16}:DISPlay:MARKer:ALL[:STATe] <char> 5-14
:CALCulate{1-16}:DISPlay:MARKer:ALL[:STATe]? 5-14
:CALCulate{1-16}:EOOE:EO4Measurment:CALCulate? 5-15
:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile <char> 5-15
:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile:SWAP[:STATe] <char> 5-16
:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile:SWAP[:STATe]? 5-16
:CALCulate{1-16}:EOOE:EO4Measurment:CHARfile? 5-15
:CALCulate{1-16}:EOOE:EO4Measurment:CONFiguration <char> 5-16
:CALCulate{1-16}:EOOE:EO4Measurment:CONFiguration? 5-16
:CALCulate{1-16}:EOOE:EO4Measurment:EOPort <char> 5-16
:CALCulate{1-16}:EOOE:EO4Measurment:EOPort? 5-16
:CALCulate{1-16}:EOOE:EO4Measurment:OEPort <char> 5-17
:CALCulate{1-16}:EOOE:EO4Measurment:OEPort? 5-17
:CALCulate{1-16}:EOOE:EOMeasurment:CALCulate? 5-17
:CALCulate{1-16}:EOOE:EOMeasurment:CALFile <string> 5-18
:CALCulate{1-16}:EOOE:EOMeasurment:CALFile? 5-18
:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile <string> 5-18
:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile:SWAP[:STATe] <char> 5-18
:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile:SWAP[:STATe]? 5-18
:CALCulate{1-16}:EOOE:EOMeasurment:CHARfile? 5-18
:CALCulate{1-16}:EOOE:EOMeasurment:EOPort <char> 5-19

:CALCulate{1-16}:EOOE:EOMeasurment:EOPort? 5-19
:CALCulate{1-16}:EOOE:GO4Measurment:CALCulate? 5-19
:CALCulate{1-16}:EOOE:GO4Measurment:CALFile <string> 5-20
:CALCulate{1-16}:EOOE:GO4Measurment:CALFile? 5-20
:CALCulate{1-16}:EOOE:GO4Measurment:CHARfile <char> 5-20
:CALCulate{1-16}:EOOE:GO4Measurment:CHARfile? 5-20
:CALCulate{1-16}:EOOE:GO4Measurment:CONFiguration <char> 5-20
:CALCulate{1-16}:EOOE:GO4Measurment:CONFiguration? 5-20
:CALCulate{1-16}:EOOE:GO4Measurment:EOPort <char> 5-21
:CALCulate{1-16}:EOOE:GO4Measurment:EOPort? 5-21
:CALCulate{1-16}:EOOE:GO4Measurment:OEPort <char> 5-21
:CALCulate{1-16}:EOOE:GO4Measurment:OEPort? 5-21
:CALCulate{1-16}:EOOE:GO4Measurment:TARGetfile <string> 5-21
:CALCulate{1-16}:EOOE:GO4Measurment:TARGetfile? 5-21
:CALCulate{1-16}:EOOE:GOMeasurment:CALCulate? 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:CALFile <string> 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:CALFile? 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:CHARfile <char> 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:CHARfile? 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:EOPort <char> 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:EOPort? 5-22
:CALCulate{1-16}:EOOE:GOMeasurment:TARGetfile <string> 5-23
:CALCulate{1-16}:EOOE:GOMeasurment:TARGetfile? 5-23
:CALCulate{1-16}:EOOE:MSGS:LIST? 5-23
:CALCulate{1-16}:EOOE:OE4Measurment:CALCulate? 5-23
:CALCulate{1-16}:EOOE:OE4Measurment:CALFile <string> 5-24
:CALCulate{1-16}:EOOE:OE4Measurment:CALFile? 5-24
:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile <string> 5-24
:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile:SWAP[:STATe] <char> 5-25
:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile:SWAP[:STATe]? 5-25
:CALCulate{1-16}:EOOE:OE4Measurment:CHARfile? 5-24
:CALCulate{1-16}:EOOE:OE4Measurment:CONFiguration <char> 5-25
:CALCulate{1-16}:EOOE:OE4Measurment:CONFiguration? 5-25
:CALCulate{1-16}:EOOE:OE4Measurment:EOPort <char> 5-25
:CALCulate{1-16}:EOOE:OE4Measurment:EOPort? 5-25
:CALCulate{1-16}:EOOE:OE4Measurment:OEPort <char> 5-26
:CALCulate{1-16}:EOOE:OE4Measurment:OEPort? 5-26
:CALCulate{1-16}:EOOE:OEMeasurment:CALCulate? 5-26
:CALCulate{1-16}:EOOE:OEMeasurment:CALFile? 5-27
:CALCulate{1-16}:EOOE:OEMeasurment:CALFile<string> 5-27
:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile <string> 5-27
:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile:SWAP[:STATe] <char> 5-27
:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile:SWAP[:STATe]? 5-27
:CALCulate{1-16}:EOOE:OEMeasurment:CHARfile? 5-27
:CALCulate{1-16}:EOOE:OEMeasurment:EOPort <char> 5-28
:CALCulate{1-16}:EOOE:OEMeasurment:EOPort? 5-28
:CALCulate{1-16}:EOOE:TYPE? 5-28
:CALCulate{1-16}:EXTRaction 5-7
:CALCulate{1-16}:EXTRaction:CALibration:INNer <string> 5-8
:CALCulate{1-16}:EXTRaction:CALibration:INNer? 5-8
:CALCulate{1-16}:EXTRaction:CALibration:OUTer <string> 5-8
:CALCulate{1-16}:EXTRaction:CALibration:OUTer? 5-8
:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:FILe <string> 5-7
:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:FILe? 5-7
:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:PORT <char> 5-8
:CALCulate{1-16}:EXTRaction:CALibration[:CALa]:PORT? 5-8
:CALCulate{1-16}:EXTRaction:ELL1:LENGth <NRf> 5-9
:CALCulate{1-16}:EXTRaction:ELL1:LENGth? 5-9
:CALCulate{1-16}:EXTRaction:ELL2:LENGth <NRf> 5-9
:CALCulate{1-16}:EXTRaction:ELL2:LENGth? 5-9
:CALCulate{1-16}:EXTRaction:S2P1filename:FILe <string> 5-10

:CALCulate{1-16}:EXTRaction:S2P1filename:FILe? 5-10
:CALCulate{1-16}:EXTRaction:S2P2filename:FILe <string> 5-10
:CALCulate{1-16}:EXTRaction:S2P2filename:FILe? 5-10
:CALCulate{1-16}:EXTRaction:SXPPortpair:PORT <char> 5-10
:CALCulate{1-16}:EXTRaction:SXPPortpair:PORT? 5-10
:CALCulate{1-16}:EXTRaction:ZERo:MATCh[:STATe] <char> 5-11
:CALCulate{1-16}:EXTRaction:ZERo:MATCh[:STATe]? 5-11
:CALCulate{1-16}:EXTRaction[:METHod]:A 5-11
:CALCulate{1-16}:EXTRaction[:METHod]:B 5-12
:CALCulate{1-16}:EXTRaction[:METHod]:C 5-12
:CALCulate{1-16}:EXTRaction[:METHod]:D 5-13
:CALCulate{1-16}:FORMat:S1P:PORT <char> 5-29
:CALCulate{1-16}:FORMat:S1P:PORT? 5-29
:CALCulate{1-16}:FORMat:S2P:PORT <char> 5-29
:CALCulate{1-16}:FORMat:S2P:PORT? 5-29
:CALCulate{1-16}:FSIMulator:N 5-37
:CALCulate{1-16}:FSIMulator:NETWork:ADD 5-30
:CALCulate{1-16}:FSIMulator:NETWork:C <NRf> 5-30
:CALCulate{1-16}:FSIMulator:NETWork:C? 5-30
:CALCulate{1-16}:FSIMulator:NETWork:CLEar 5-30
:CALCulate{1-16}:FSIMulator:NETWork:COUNt? 5-31
:CALCulate{1-16}:FSIMulator:NETWork:DIELectric <NRf> 5-31
:CALCulate{1-16}:FSIMulator:NETWork:DIELectric? 5-31
:CALCulate{1-16}:FSIMulator:NETWork:FREQuency <NRf> 5-31
:CALCulate{1-16}:FSIMulator:NETWork:FREQuency? 5-31
:CALCulate{1-16}:FSIMulator:NETWork:L <NRf> 5-32
:CALCulate{1-16}:FSIMulator:NETWork:L? 5-32
:CALCulate{1-16}:FSIMulator:NETWork:LENGth <NRf> 5-32
:CALCulate{1-16}:FSIMulator:NETWork:LENGth? 5-32
:CALCulate{1-16}:FSIMulator:NETWork:LOSS <NRf> 5-32
:CALCulate{1-16}:FSIMulator:NETWork:LOSS? 5-32
:CALCulate{1-16}:FSIMulator:NETWork:MODe <char> 5-33
:CALCulate{1-16}:FSIMulator:NETWork:MODe? 5-33
:CALCulate{1-16}:FSIMulator:NETWork:PORT <char> 5-33
:CALCulate{1-16}:FSIMulator:NETWork:PORT? 5-33
:CALCulate{1-16}:FSIMulator:NETWork:R <NRf> 5-33
:CALCulate{1-16}:FSIMulator:NETWork:R? 5-33
:CALCulate{1-16}:FSIMulator:NETWork:S2P <string> 5-34
:CALCulate{1-16}:FSIMulator:NETWork:S2P? 5-34
:CALCulate{1-16}:FSIMulator:NETWork:SWAPs2p <char> 5-34
:CALCulate{1-16}:FSIMulator:NETWork:SWAPs2p? 5-34
:CALCulate{1-16}:FSIMulator:NETWork:TYPe <char> 5-34
:CALCulate{1-16}:FSIMulator:NETWork:TYPe? 5-34
:CALCulate{1-16}:FSIMulator:NETWork:Z0 <NRf> 5-35
:CALCulate{1-16}:FSIMulator:NETWork:Z0? 5-35
:CALCulate{1-16}:FSIMulator:NETWork[:STATe] <char> 5-35
:CALCulate{1-16}:FSIMulator:NETWork[:STATe]? 5-35
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:C <NRf> 5-36
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:C? 5-36
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:DELete 5-36
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:DIELectric <NRf> 5-36
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:DIELectric? 5-36
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:FREQuency <NRf> 5-37
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:FREQuency? 5-37
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:L <NRf> 5-37
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LENGth <NRf> 5-37
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LENGth? 5-37
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LOSS <NRf> 5-38
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:LOSS? 5-38
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:MODe <char> 5-38
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:MODe? 5-38

:CALCulate{1-16}:FSIMulator:NETWork{1-50}:PORT <char> 5-38
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:PORT? 5-38
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:R <NRf> 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:R? 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:S2P <string> 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:S2P? 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:SWAPs2p <char> 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:SWAPs2p? 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:TYPe <char> 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:TYPe? 5-39
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:Z0 <NRf> 5-40
:CALCulate{1-16}:FSIMulator:NETWork{1-50}:Z0? 5-40
:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:R0 <NRf> 5-41
:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:R0? 5-41
:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:X0 <NRf> 5-41
:CALCulate{1-16}:IMPedance:TRANsformation:PORT{1-2}:X0? 5-41
:CALCulate{1-16}:IMPedance:TRANsformation[:STATe] 5-42
:CALCulate{1-16}:IMPedance:TRANsformation[:STATe]? 5-42
:CALCulate{1-16}:MARKer:COUPle <char> 5-43
:CALCulate{1-16}:MARKer:COUPle? 5-43
:CALCulate{1-16}:MARKer:TABle[:STATe] <char> 5-43
:CALCulate{1-16}:MARKer:TABle[:STATe]? 5-43
:CALCulate{1-16}:PARameter:COUNt <NRf> 5-44
:CALCulate{1-16}:PARameter:COUNt? 5-44
:CALCulate{1-16}:PARameter:SELect? 5-46
:CALCulate{1-16}:PARameter(1-16):DATA:FMEMory? 5-45
:CALCulate{1-16}:PARAmeter(1-16):DATA:SDATa? 5-45
:CALCulate{1-16}:PARAmeter(1-16):DATA:SMEMory? 5-45
:CALCulate{1-16}:PARameter{1-16}:DATA:FDATa? 5-44
:CALCulate{1-16}:PARameter{1-16}:DEFine <char1> | <char1>,<char2> | <char1>,<char2>,<char3>,<char4> 5-46
:CALCulate{1-16}:PARameter{1-16}:DEFine? 5-46
:CALCulate{1-16}:PARameter{1-16}:FORMat <char> 5-48
:CALCulate{1-16}:PARameter{1-16}:FORMat? 5-48
:CALCulate{1-16}:PARameter{1-16}:IMPedance:LC[:STATe] <char> 5-49
:CALCulate{1-16}:PARameter{1-16}:IMPedance:LC[:STATe]? 5-49
:CALCulate{1-16}:PARameter{1-16}:MARKer:ACTivate? 5-50
:CALCulate{1-16}:PARameter{1-16}:MARKer:DISCrete <char> 5-50
:CALCulate{1-16}:PARameter{1-16}:MARKer:DISCrete? 5-50
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:ACTivate 5-51
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:X <NRf> 5-51
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:X? 5-51
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}:Y? 5-51
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}[:STATe] <char> 5-52
:CALCulate{1-16}:PARameter{1-16}:MARKer{1-13}[:STATe]? 5-52
:CALCulate{1-16}:PARameter{1-16}:MEA:DEFine <NRf> 5-49
:CALCulate{1-16}:PARameter{1-16}:MEA:DEFine? 5-49
:CALCulate{1-16}:PARameter{1-16}:MSTatistics <char> 5-53
:CALCulate{1-16}:PARameter{1-16}:MSTatistics:DATA? 5-53
:CALCulate{1-16}:PARameter{1-16}:MSTatistics:DATA2? 5-54
:CALCulate{1-16}:PARameter{1-16}:MSTatistics? 5-53
:CALCulate{1-16}:PARameter{1-16}:SELect 5-55
:CALCulate{1-16}:POLar:ANGLe:STARt <NRf> 5-56
:CALCulate{1-16}:POLar:ANGLe:STARt? 5-56
:CALCulate{1-16}:POLar:ANGLe:STOP <NRf> 5-56
:CALCulate{1-16}:POLar:ANGLe:STOP? 5-56
:CALCulate{1-16}:POLar:CHARt <char> 5-57
:CALCulate{1-16}:POLar:CHARt? 5-57
:CALCulate{1-16}:PROCessing:ORDer:GRPDelay <char> 5-58
:CALCulate{1-16}:PROCessing:ORDer:GRPDelay? 5-58
:CALCulate{1-16}:PROCessing:ORDer:REFPlane <char> 5-58
:CALCulate{1-16}:PROCessing:ORDer:REFPlane? 5-58

:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric <char> 5-59
:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric:OTHer <NRf> 5-59
:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric:OTHer? 5-59
:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric:VALue? 5-60
:CALCulate{1-16}:REFerence:EXTension:COAXial:DIELectric? 5-59
:CALCulate{1-16}:REFerence:EXTension:LINE <char> 5-60
:CALCulate{1-16}:REFerence:EXTension:LINE? 5-60
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:AUTOmatic 5-60
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:DISTance <NRf> 5-60
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:DISTance? 5-60
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:LOSS <NRf> 5-61
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:LOSS? 5-61
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:PHAse <NRf> 5-61
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:PHAse? 5-61
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:TIMe <NRf> 5-61
:CALCulate{1-16}:REFerence:EXTension:PORT{1-2}:TIMe? 5-61
:CALCulate{1-16}:RFRanging: <char> 5-62
:CALCulate{1-16}[:SELected]:ALTernate:TRACe:NAMe "user defined name" 5-4
:CALCulate{1-16}[:SELected]:ALTernate:TRACe:NAMe? 5-4
:CALCulate{1-16}[:SELected]:CONVersion:FUNCtion <char> 5-63
:CALCulate{1-16}[:SELected]:CONVersion:FUNCtion? 5-63
:CALCulate{1-16}[:SELected]:CONVersion[:STATe] <char> 5-64
:CALCulate{1-16}[:SELected]:CONVersion[:STATe]? 5-64
:CALCulate{1-16}[:SELected]:DATA:FDATa <block> 5-65
:CALCulate{1-16}[:SELected]:DATA:FDATa? 5-65
:CALCulate{1-16}[:SELected]:DATA:FMEMory <block> 5-66
:CALCulate{1-16}[:SELected]:DATA:FMEMory? 5-66
:CALCulate{1-16}[:SELected]:DATA:SDATa <Block> 5-66
:CALCulate{1-16}[:SELected]:DATA:SDATa? 5-66
:CALCulate{1-16}[:SELected]:DATA:SMEMory <Block> 5-66
:CALCulate{1-16}[:SELected]:DATA:SMEMory? 5-66
:CALCulate{1-16}[:SELected]:FORMat <char> 5-67
:CALCulate{1-16}[:SELected]:FORMat? 5-67
:CALCulate{1-16}[:SELected]:IMPedance:LC[:STATe] <char> 5-68
:CALCulate{1-16}[:SELected]:IMPedance:LC[:STATe]? 5-68
:CALCulate{1-16}[:SELected]:LIMit:ALARm <char> 5-69
:CALCulate{1-16}[:SELected]:LIMit:ALARm? 5-69
:CALCulate{1-16}[:SELected]:LIMit:DATA <block> 5-70
:CALCulate{1-16}[:SELected]:LIMit:DATA? 5-70
:CALCulate{1-16}[:SELected]:LIMit:DISPlay[:STATe] <char> 5-70
:CALCulate{1-16}[:SELected]:LIMit:DISPlay[:STATe]? 5-70
:CALCulate{1-16}[:SELected]:LIMit:FAIL? 5-70
:CALCulate{1-16}[:SELected]:LIMit:OFF 5-70
:CALCulate{1-16}[:SELected]:LIMit:REPort:POINt? 5-71
:CALCulate{1-16}[:SELected]:LIMit:REPort? 5-71
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:ADD {No argument} | {<char>} | {<char>,<NRf>,<NRf>} 5-71
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:CLEar 5-72
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:COUNt? 5-72
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:DEFine <NRf>,<NRf> | <NRf>,<NRf>,<NRf>,<NRf> 5-72
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:DEFine? 5-72
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:TYPe <char> 5-73
:CALCulate{1-16}[:SELected]:LIMit:SEGMent:TYPe? 5-73
:CALCulate{1-16}[:SELected]:LIMit[:STATe] <char> 5-72
:CALCulate{1-16}[:SELected]:LIMit[:STATe]? 5-72
:CALCulate{1-16}[:SELected]:MARKer:ACTivate? 5-73
:CALCulate{1-16}[:SELected]:MARKer:ALL[:STATe] <char> 5-74
:CALCulate{1-16}[:SELected]:MARKer:MOVe:CENTer 5-74
:CALCulate{1-16}[:SELected]:MARKer:MOVe:REFMarker 5-74
:CALCulate{1-16}[:SELected]:MARKer:MOVe:STARt 5-74
:CALCulate{1-16}[:SELected]:MARKer:MOVe:STOP 5-75
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch 5-75

:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:EXCursion <NRf> 5-75
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:EXCursion? 5-75
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:POLarity <char> 5-75
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:POLarity? 5-75
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:THREshold <NRf> 5-76
:CALCulate{1-16}[:SELected]:MARKer:MPSEArch:THREshold? 5-76
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch 5-76
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TARget <NRf> 5-76
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TARget? 5-76
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TRANsition <char> 5-76
:CALCulate{1-16}[:SELected]:MARKer:MTSEArch:TRANsition? 5-76
:CALCulate{1-16}[:SELected]:MARKer:OFF 5-77
:CALCulate{1-16}[:SELected]:MARKer:PSEArch <char> 5-77
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:EXCursion <NRf> 5-77
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:EXCursion? 5-77
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:POLarity <char> 5-77
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:POLarity? 5-77
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:THREshold <NRf> 5-78
:CALCulate{1-16}[:SELected]:MARKer:PSEArch:THREshold? 5-78
:CALCulate{1-16}[:SELected]:MARKer:PSEArch? 5-77
:CALCulate{1-16}[:SELected]:MARKer:SEArch <char> 5-78
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:DATA? 5-78
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:DEFine <NRf> 5-79
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:DEFine? 5-79
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:RTYPe <char> 5-79
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:RTYPe? 5-79
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:RVALue? 5-79
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:HIGH <NRf> 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:HIGH? 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:LOW <NRf> 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe:LOW? 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe[:STATe] <char> 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SHAPe[:STATe]? 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth:SSTart? 5-80
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth[:STATe] <char> 5-81
:CALCulate{1-16}[:SELected]:MARKer:SEArch:BANDwidth[:STATe]? 5-81
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:DATA? 5-81
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:DEFine <NRf> 5-81
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:DEFine? 5-81
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RTYPe <char> 5-82
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RTYPe? 5-82
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RVALue <char> 5-82
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:RVALue? 5-82
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:HIGH <NRf> 5-82
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:HIGH? 5-82
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:LOW <NRf> 5-83
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe:LOW? 5-83
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe[:STATe] <char> 5-83
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SHAPe[:STATe]? 5-83
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SSTart <char> 5-83
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh:SSTart? 5-83
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh[:STATe] <char> 5-84
:CALCulate{1-16}[:SELected]:MARKer:SEArch:NOTCh[:STATe]? 5-84
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:ALL[:STATe] <char> 5-84
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:ALL[:STATe]? 5-84
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STARt:X <NRf> 5-84
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STARt:X? 5-84
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STOP:X <NRf> 5-85
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe:STOP:X? 5-85
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe[:STATe] <char> 5-85
:CALCulate{1-16}[:SELected]:MARKer:SEArch:RANGe[:STATe]? 5-85

:CALCulate{1-16}[:SELected]:MARKer:SEArch:TRACKing[:STATe] <char> 5-85
:CALCulate{1-16}[:SELected]:MARKer:SEArch:TRACKing[:STATe]? 5-85
:CALCulate{1-16}[:SELected]:MARKer:SEArch? 5-78
:CALCulate{1-16}[:SELected]:MARKer:SET:CENTer 5-85
:CALCulate{1-16}[:SELected]:MARKer:SET:REFLevel 5-86
:CALCulate{1-16}[:SELected]:MARKer:SET:STARt 5-86
:CALCulate{1-16}[:SELected]:MARKer:SET:STOP 5-86
:CALCulate{1-16}[:SELected]:MARKer:TSEArch <char> 5-86
:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TARget <NRf> 5-86
:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TARget? 5-86
:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TRANsition <char> 5-87
:CALCulate{1-16}[:SELected]:MARKer:TSEArch:TRANsition? 5-87
:CALCulate{1-16}[:SELected]:MARKer:TSEArch? 5-86
:CALCulate{1-16}[:SELected]:MARKer{1-13}:ACTivate 5-88
:CALCulate{1-16}[:SELected]:MARKer{1-13}:MOVe <char> 5-88
:CALCulate{1-16}[:SELected]:MARKer{1-13}:SET <char> 5-89
:CALCulate{1-16}[:SELected]:MARKer{1-13}:X <NRf> 5-89
:CALCulate{1-16}[:SELected]:MARKer{1-13}:X? 5-89
:CALCulate{1-16}[:SELected]:MARKer{1-13}:Y? 5-89
:CALCulate{1-16}[:SELected]:MARKer{1-13}[:STATe] <char> 5-90
:CALCulate{1-16}[:SELected]:MARKer{1-13}[:STATe]? 5-90
:CALCulate{1-16}[:SELected]:MATH:DISPlay <char> 5-91
:CALCulate{1-16}[:SELected]:MATH:DISPlay? 5-91
:CALCulate{1-16}[:SELected]:MATH:FUNCtion <char> 5-92
:CALCulate{1-16}[:SELected]:MATH:FUNCtion? 5-92
:CALCulate{1-16}[:SELected]:MATH:INTErtrace:FUNCtion <char> 5-92
:CALCulate{1-16}[:SELected]:MATH:INTErtrace:FUNCtion? 5-92
:CALCulate{1-16}[:SELected]:MATH:INTErtrace:OPERand{1-2}:DEFine <char1>, <char2> 5-92
:CALCulate{1-16}[:SELected]:MATH:INTErtrace:OPERand{1-2}:DEFine? 5-92
:CALCulate{1-16}[:SELected]:MATH:INTErtrace[:STATe] <char> 5-93
:CALCulate{1-16}[:SELected]:MATH:INTErtrace[:STATe]? 5-93
:CALCulate{1-16}[:SELected]:MATH:MEMorize 5-93
:CALCulate{1-16}[:SELected]:MDATA:FDATa <block> 5-94
:CALCulate{1-16}[:SELected]:MDATA:FDATa? 5-94
:CALCulate{1-16}[:SELected]:MDATA:SDATa <block> 5-94
:CALCulate{1-16}[:SELected]:MDATA:SDATa? 5-94
:CALCulate{1-16}[:SELected]:SMOothing:APERture <NRf> 5-95
:CALCulate{1-16}[:SELected]:SMOothing:APERture? 5-95
:CALCulate{1-16}[:SELected]:SMOothing[:STATe] <char> 5-95
:CALCulate{1-16}[:SELected]:SMOothing[:STATe]? 5-95
:CALCulate{1-16}[:SELected]:TDATA:FDATa <block> 5-96
:CALCulate{1-16}[:SELected]:TDATA:FDATa? 5-96
:CALCulate{1-16}[:SELected]:TDATA:SDATa <block> 5-97
:CALCulate{1-16}[:SELected]:TDATA:SDATa? 5-97
:CALCulate{1-16}[:SELected]:TRANsform:TIME:ALIAsfree? 5-98
:CALCulate{1-16}[:SELected]:TRANsform:TIME:CENTer <NRf> 5-98
:CALCulate{1-16}[:SELected]:TRANsform:TIME:CENTer? 5-98
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm <char> 5-99
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm:OTHer <NRf> 5-99
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm:OTHer? 5-99
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DCTerm? 5-99
:CALCulate{1-16}[:SELected]:TRANsform:TIME:DISTance? 5-99
:CALCulate{1-16}[:SELected]:TRANsform:TIME:EXTrapolate <char> 5-100
:CALCulate{1-16}[:SELected]:TRANsform:TIME:EXTrapolate? 5-100
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:CENTer <NRf> 5-100
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:CENTer? 5-100
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:DCGamma <NRf> 5-100
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:DCGamma? 5-100
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:KBBeta <NRf> 5-101
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:KBBeta? 5-101
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:NOTCh[:STATe] <char> 5-101

:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:NOTCh[:STATe]? 5-101
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SHAPe <char> 5-101
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SHAPe? 5-101
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SPAN <NRf> 5-102
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:SPAN? 5-102
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STARt <NRf> 5-102
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STARt? 5-102
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STOP <NRf> 5-102
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE:STOP? 5-102
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE[:STATe] <char> 5-103
:CALCulate{1-16}[:SELected]:TRANsform:TIME:GATE[:STATe]? 5-103
:CALCulate{1-16}[:SELected]:TRANsform:TIME:IMPUlsewidth? 5-103
:CALCulate{1-16}[:SELected]:TRANsform:TIME:RESPonse <char> 5-103
:CALCulate{1-16}[:SELected]:TRANsform:TIME:RESPonse? 5-103
:CALCulate{1-16}[:SELected]:TRANsform:TIME:SPAN <NRf> 5-104
:CALCulate{1-16}[:SELected]:TRANsform:TIME:SPAN? 5-104
:CALCulate{1-16}[:SELected]:TRANsform:TIME:STARt <NRf> 5-104
:CALCulate{1-16}[:SELected]:TRANsform:TIME:STARt? 5-104
:CALCulate{1-16}[:SELected]:TRANsform:TIME:STOP <NRf> 5-104
:CALCulate{1-16}[:SELected]:TRANsform:TIME:STOP? 5-104
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TIME? 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TRIP <char> 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TRIP? 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TYPe <char> 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:TYPe? 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:UNIt <char> 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:UNIt? 5-105
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:DCGamma <NRf> 5-106
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:DCGamma? 5-106
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:KBBeta <NRf> 5-106
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:KBBeta? 5-106
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:SHAPe <char> 5-106
:CALCulate{1-16}[:SELected]:TRANsform:TIME:WINDow{1-16}:SHAPe? 5-106
:DISPlay:ANNotation:FREQuency <char> 5-107
:DISPlay:ANNotation:FREQuency <char> 5-107
:DISPlay:COLor:INVert:BACK <NRf>, <NRf>, <NRf> 5-108
:DISPlay:COLor:INVert:BACK? 5-108
:DISPlay:COLor:INVert:GRATicule:MAIN <NRf>, <NRf>, <NRf> 5-108
:DISPlay:COLor:INVert:GRATicule:MAIN? 5-108
:DISPlay:COLor:INVert:GRATicule:SUB <NRf>, <NRf>, <NRf> 5-108
:DISPlay:COLor:INVert:GRATicule:SUB? 5-108
:DISPlay:COLor:INVert:LIMit <NRf>, <NRf>, <NRf> 5-109
:DISPlay:COLor:INVert:LIMit? 5-109
:DISPlay:COLor:INVert:TRACe{1-16}:DATA <NRf>, <NRf>, <NRf> 5-109
:DISPlay:COLor:INVert:TRACe{1-16}:DATA? 5-109
:DISPlay:COLor:INVert:TRACe{1-16}:MEMory <NRf>, <NRf>, <NRf> 5-109
:DISPlay:COLor:INVert:TRACe{1-16}:MEMory? 5-109
:DISPlay:COLor:INVert[:STATe] <char> 5-110
:DISPlay:COLor:INVert[:STATe]? 5-110
:DISPlay:COLor:NORMal:BACK <NRf>, <NRf>, <NRf> 5-110
:DISPlay:COLor:NORMal:BACK? 5-110
:DISPlay:COLor:NORMal:GRATicule:MAIN <NRf>, <NRf>, <NRf> 5-110
:DISPlay:COLor:NORMal:GRATicule:MAIN? 5-110
:DISPlay:COLor:NORMal:GRATicule:SUB <NRf>, <NRf>, <NRf> 5-111
:DISPlay:COLor:NORMal:GRATicule:SUB? 5-111
:DISPlay:COLor:NORMal:LIMit <NRf>, <NRf>, <NRf> 5-111
:DISPlay:COLor:NORMal:LIMit? 5-111
:DISPlay:COLor:NORMal:TRACe{1-16}:DATA <NRf>, <NRf>, <NRf> 5-111
:DISPlay:COLor:NORMal:TRACe{1-16}:DATA? 5-111
:DISPlay:COLor:NORMal:TRACe{1-16}:MEMory <NRf>, <NRf>, <NRf> 5-112
:DISPlay:COLor:NORMal:TRACe{1-16}:MEMory? 5-112

:DISPlay:COLor:RESet . . . 5-112
:DISPlay:COUNt <NRf> . . . 5-113
:DISPlay:COUNt? . . . 5-113
:DISPlay:FSIGn[:STATe] <char> . . . 5-113
:DISPlay:FSIGn[:STATe]? . . . 5-113
:DISPlay:RECONFig:Clear . . . 5-113
:DISPlay:RECONFig:COUNt? <External Serial Number> . . . 5-114
:DISPlay:RECONFig:STart . . . 5-114
:DISPlay:SIZe <char> . . . 5-113
:DISPlay:SIZe? . . . 5-113
:DISPlay:WINDow:ACTivate? . . . 5-114
:DISPlay:WINDow{1-16}:ACTivate . . . 5-114
:DISPlay:WINDow{1-16}:SPLit <char> . . . 5-115
:DISPlay:WINDow{1-16}:SPLit? . . . 5-115
:DISPlay:WINDow{1-16}:TITLe <string> . . . 5-116
:DISPlay:WINDow{1-16}:TITLe:STATe <char> . . . 5-116
:DISPlay:WINDow{1-16}:TITLe:STATe? . . . 5-116
:DISPlay:WINDow{1-16}:TITLe? . . . 5-116
:DISPlay:WINDow{1-16}:TRACe{1-16}:SIZe <char> . . . 5-116
:DISPlay:WINDow{1-16}:TRACe{1-16}:SIZe? . . . 5-116
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:AUTO . . . 5-117
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV <NRf> . . . 5-117
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV? . . . 5-117
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV2 <NRf> . . . 5-118
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PDIV2? . . . 5-118
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:OFFSet <NRf> . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:OFFSet? . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:WRAPping[:STATe] <char> . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHASe:WRAPping[:STATe]? . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHOFF <NRf> . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:PHOFF? . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV <NRf> . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV? . . . 5-119
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV2 <NRf> . . . 5-120
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RLEV2? . . . 5-120
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS <NRf> . . . 5-120
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS? . . . 5-120
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS2 <NRf> . . . 5-120
:DISPlay:WINDow{1-16}:TRACe{1-16}:Y:RPOS2? . . . 5-120
:DISPlay:WINDow{1-16}:Y:AUTO . . . 5-121
:DISPlay:WINDow{1-16}:Y:NDIVisions <NRf> . . . 5-115
:DISPlay:WINDow{1-16}:Y:NDIVisions? . . . 5-115
:DISPlay:Y:AUTO . . . 5-121
:DISPlay:Y:NDIVisions . . . 5-121
:FORMat:BORDer <char> . . . 5-122
:FORMat:BORDer? . . . 5-122
:FORMat:DATA <char> . . . 5-122
:FORMat:DATA:HEADing[:STATe] <char> . . . 5-123
:FORMat:DATA:HEADing[:STATe]? . . . 5-123
:FORMat:DATA? . . . 5-122
:FORMat:SNP:FREQuency <char> . . . 5-123
:FORMat:SNP:FREQuency? . . . 5-123
:FORMat:SNP:PARameter <char> . . . 5-123
:FORMat:SNP:PARameter? . . . 5-123
:HCOPy:DEVice:ID <String> . . . 5-125
:HCOPy:DEVice:ID:STATe <char> . . . 5-126
:HCOPy:DEVice:ID:STATe? . . . 5-126
:HCOPy:DEVice:ID? . . . 5-125
:HCOPy:IMAGe <char> . . . 5-126
:HCOPy:IMAGe? . . . 5-126
:HCOPy:MODel <String> . . . 5-126

:HCOPy:MODel:STATe? 5-127
:HCOPy:MODel:STATe<char> 5-127
:HCOPy:MODel? 5-126
:HCOPy:OPERator:COMMent:STATe <char> 5-127
:HCOPy:OPERator:COMMent:STATe? 5-127
:HCOPy:OPERator:COMMent? 5-127
:HCOPy:OPERator:COMMent<String> 5-127
:HCOPy:OPERator:NAMe <String> 5-128
:HCOPy:OPERator:NAMe:STATe <char> 5-128
:HCOPy:OPERator:NAMe:STATe? 5-128
:HCOPy:OPERator:NAMe? 5-128
:HCOPy:PRINt:DATe:TIMe:STATe <char> 5-128
:HCOPy:PRINt:DATe:TIMe:STATe? 5-128
:HCOPy:PRINt:HEADers:STATe <char> 5-129
:HCOPy:PRINt:HEADers:STATe? 5-129
:HCOPy:PRINt:LOGO:STATe <char> 5-129
:HCOPy:PRINt:LOGO:STATe? 5-129
:HCOPy:PRINt:LOGO:TYPe <char> 5-129
:HCOPy:PRINt:LOGO:TYPe? 5-129
:HCOPy:PRINt:TYPe <char> 5-129
:HCOPy:PRINt:TYPe? 5-129
:HCOPy[:IMMediate] 5-125
:MMEMory:CATalog? {<string>} 5-130
:MMEMory:COPY <string1>, <string2> 5-130
:MMEMory:DELete <string> 5-130
:MMEMory:LOAD <string> 5-131
:MMEMory:LOAD:CKIT <string> 5-131
:MMEMory:LOAD:FSEGMent <string> 5-131
:MMEMory:LOAD:ISEGMent <string> 5-131
:MMEMory:LOAD:LIMit <string> 5-132
:MMEMory:LOAD:MDATA <string> 5-132
:MMEMory:MDIRectory <string> 5-132
:MMEMory:RDIRectory <string> 5-132
:MMEMory:STORe <string> 5-132
:MMEMory:STORe:FSEGMent <string> 5-133
:MMEMory:STORe:IMAGe <string> 5-133
:MMEMory:STORe:ISEGMent <string> 5-133
:MMEMory:STORe:LIMit <string> 5-133
:MMEMory:STORe:MDATA <string> 5-133
:MMEMory:TRANsfer <string>, <block> 5-134
:MMEMory:TRANsfer? 5-134
:MMEMory:WRITE:CKIT <char1>, <char2>, <char3>, <string> 5-134
:SENSe{1-16}:ABORtcal 5-136
:SENSe{1-16}:AVERage:CLEar 5-136
:SENSe{1-16}:AVERage:COUNt <NRf> 5-137
:SENSe{1-16}:AVERage:COUNt? 5-137
:SENSe{1-16}:AVERage:SWEep? 5-137
:SENSe{1-16}:AVERage:TYPe <char> 5-137
:SENSe{1-16}:AVERage:TYPe? 5-137
:SENSe{1-16}:AVERage[:STATe] <char> 5-137
:SENSe{1-16}:AVERage[:STATe]? 5-137
:SENSe{1-16}:BANDwidth[:RESolution] <NRf> 5-138
:SENSe{1-16}:BANDwidth[:RESolution]? 5-138
:SENSe{1-16}:BWIDth[:RESolution] <NRf> 5-139
:SENSe{1-16}:BWIDth[:RESolution]? 5-139
:SENSe{1-16}:CORRection:COEFficient <char>, <block> 5-140
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:1P2PF 5-141
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:1P2PR 5-141
:SENSe{1-16}:CORRection:COEFficient:PORT{1-2}:FULL1 5-143
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:FULL2 5-141
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:FULLB 5-141

:SENSe{1-16}:CORRection:COEFficient:PORT{1-2}:RESP1 5-143
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:RESPB 5-142
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:TFRB 5-142
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:TFRF 5-142
:SENSe{1-16}:CORRection:COEFficient:PORT{12}:TFRR 5-142
:SENSe{1-16}:CORRection:COEFficient? <char> 5-140
:SENSe{1-16}:CORRection:COEFficient[:METHod]:1P2PF 5-144
:SENSe{1-16}:CORRection:COEFficient[:METHod]:1P2PR 5-145
:SENSe{1-16}:CORRection:COEFficient[:METHod]:FULL1 5-145
:SENSe{1-16}:CORRection:COEFficient[:METHod]:FULL2 5-145
:SENSe{1-16}:CORRection:COEFficient[:METHod]:FULLB 5-145
:SENSe{1-16}:CORRection:COEFficient[:METHod]:RESP1 5-146
:SENSe{1-16}:CORRection:COEFficient[:METHod]:RESPB 5-146
:SENSe{1-16}:CORRection:COEFficient[:METHod]:TFRB 5-146
:SENSe{1-16}:CORRection:COEFficient[:METHod]:TFRF 5-147
:SENSe{1-16}:CORRection:COEFficient[:METHod]:TFRR 5-147
:SENSe{1-16}:CORRection:COLLect:ECAL:AUTOmatic:ORIentation[:STATe] <char> 5-148
:SENSe{1-16}:CORRection:COLLect:ECAL:AUTOmatic:ORIentation[:STATe]? 5-148
:SENSe{1-16}:CORRection:COLLect:ECAL:BEGin? 5-148
:SENSe{1-16}:CORRection:COLLect:ECAL:CONTinue? 5-149
:SENSe{1-16}:CORRection:COLLect:ECAL:ORIentation <char> 5-150
:SENSe{1-16}:CORRection:COLLect:ECAL:ORIentation? 5-150
:SENSe{1-16}:CORRection:COLLect:ECAL[:CALa]:TRUEthru <char> 5-150
:SENSe{1-16}:CORRection:COLLect:ECAL[:CALa]:TRUEthru? 5-150
:SENSe{1-16}:CORRection:COLLect:FULL2 5-193
:SENSe{1-16}:CORRection:COLLect:FULLB 5-193
:SENSe{1-16}:CORRection:COLLect:METHod <char> 5-151
:SENSe{1-16}:CORRection:COLLect:METHod? 5-151
:SENSe{1-16}:CORRection:COLLect:MICrostrip:DIELectric <NRf> 5-152
:SENSe{1-16}:CORRection:COLLect:MICrostrip:DIELectric? 5-152
:SENSe{1-16}:CORRection:COLLect:MICrostrip:EFFective <NRf> 5-153
:SENSe{1-16}:CORRection:COLLect:MICrostrip:EFFective? 5-153
:SENSe{1-16}:CORRection:COLLect:MICrostrip:KIT <char> 5-153
:SENSe{1-16}:CORRection:COLLect:MICrostrip:KIT? 5-153
:SENSe{1-16}:CORRection:COLLect:MICrostrip:PORT{1-2}:CONNector <char> 5-154
:SENSe{1-16}:CORRection:COLLect:MICrostrip:PORT{1-2}:CONNector? 5-154
:SENSe{1-16}:CORRection:COLLect:MICrostrip:THICKness <NRf> 5-154
:SENSe{1-16}:CORRection:COLLect:MICrostrip:THICKness? 5-154
:SENSe{1-16}:CORRection:COLLect:MICrostrip:WIDth <NRf> 5-155
:SENSe{1-16}:CORRection:COLLect:MICrostrip:WIDth? 5-155
:SENSe{1-16}:CORRection:COLLect:MICrostrip:Z0 <NRf> 5-155
:SENSe{1-16}:CORRection:COLLect:MICrostrip:Z0? 5-155
:SENSe{1-16}:CORRection:COLLect:MULTiple:THRu <char> {<char>,<char>,<char>,<char>,<char>} 5-156
:SENSe{1-16}:CORRection:COLLect:MULTiple:THRu? 5-156
:SENSe{1-16}:CORRection:COLLect:PORT{1 | 2 | 12}:RESP1 5-157
:SENSe{1-16}:CORRection:COLLect:PORT{1|2|12}:FULL1 5-157
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:CONNector <char> 5-161
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:CONNector? 5-161
:SENSe{1-16}:CORRection:COLLect:PORT{12}:ISOL 5-158
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD 5-162
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:SELect <char> 5-162
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:SELect? 5-162
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:TYPe <char> 5-163
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD:TYPe? 5-163
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C0 <NRf> 5-163
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C0? 5-163
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C1 <NRf> 5-163
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C1? 5-163
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C2 <NRf> 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C2? 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C3 <NRf> 5-164

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:C3? 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L0 <NRf> 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L0? 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L1 <NRf> 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L1? 5-164
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L2 <NRf> 5-165
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L2? 5-165
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L3 <NRf> 5-165
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:L3? 5-165
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:LABEL <string> 5-165
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:LABEL? 5-165
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF1 <NRf> 5-166
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF1? 5-166
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF2 <NRf> 5-166
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF2? 5-166
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF3 <NRf> 5-166
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFF3? 5-166
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFFS <NRf> 5-167
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:OFFS? 5-167
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:R <NRf> 5-167
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:R? 5-167
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:SERial <string> 5-167
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:SERial? 5-167
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:Z0 <NRf> 5-168
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD1:Z0? 5-168
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C0 <NRf> 5-168
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C0? 5-168
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C1 <NRf> 5-168
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C1? 5-168
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C2 <NRf> 5-169
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C2? 5-169
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C3 <NRf> 5-169
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:C3? 5-169
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L0 <NRf> 5-169
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L0? 5-169
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L1 <NRf> 5-170
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L1? 5-170
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L2 <NRf> 5-170
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L2? 5-170
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L3 <NRf> 5-170
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:L3? 5-170
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:LABEL <string> 5-171
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:LABEL? 5-171
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF1 <NRf> 5-171
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF1? 5-171
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF2 <NRf> 5-171
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF2? 5-171
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF3 <NRf> 5-172
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFF3? 5-172
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFFS <NRf> 5-172
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:OFFS? 5-172
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:R <NRf> 5-172
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:R? 5-172
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:SERial <string> 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:SERial? 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:Z0 <NRf> 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:LOAD2:Z0? 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C0 <NRf> 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C0? 5-173
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C1 <NRF> 5-174

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C1? 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C2 <NRf> 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C2? 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C3 <NRf> 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:C3? 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:LABEL <string> 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:LABEL? 5-174
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:OFFS <NRf> 5-175
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:OFFS? 5-175
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:SERial <string> 5-175
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:OPEN:SERial? 5-175
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:REFLection:COMPonent <char> 5-175
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:REFLection:COMPonent? 5-175
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT 5-176
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L0 <NRf> 5-176
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L0? 5-176
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L1 <NRf> 5-176
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L1? 5-176
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L2 <NRf> 5-177
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L2? 5-177
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L3 <NRf> 5-177
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:L3? 5-177
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:LABEL <string> 5-177
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:LABEL? 5-177
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:OFFS <NRf> 5-178
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:OFFS? 5-178
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:SERial <string> 5-178
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT:SERial? 5-178
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1 5-178
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L0 <NRf> 5-179
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L0? 5-179
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L1 <NRf> 5-179
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L1? 5-179
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L2 <NRf> 5-179
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L2? 5-179
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L3 <NRf> 5-180
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:L3? 5-180
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:LABEL <string> 5-180
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:LABEL? 5-180
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:OFFS <NRf> 5-180
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:OFFS? 5-180
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:SERial <string> 5-181
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT1:SERial? 5-181
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2 5-181
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L0 <NRf> 5-181
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L0? 5-181
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L1 <NRf> 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L1? 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L2 <NRf> 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L2? 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L3 <NRf> 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:L3? 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:LABEL <string> 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:LABEL? 5-182
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:OFFS <NRf> 5-183
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:OFFS? 5-183
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:SERial <string> 5-183
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT2:SERial? 5-183
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3 5-183
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L0 <NRf> 5-184
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L0? 5-184

:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L1 <NRf> 5-184
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L1? 5-184
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L2 <NRf> 5-184
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L2? 5-184
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L3 <NRf> 5-185
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:L3? 5-185
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:LABEL <string> 5-185
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:LABEL? 5-185
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:OFFS <NRf> 5-185
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:OFFS? 5-185
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:SERial <string> 5-186
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SHORT3:SERial? 5-186
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD1 5-186
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD2 5-186
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD3 5-186
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD4 5-186
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD5 5-187
:SENSe{1-16}:CORRection:COLLect:PORT{1-2}:SLOAD6 5-187
:SENSe{1-16}:CORRection:COLLect:PORT{12}:TFRB 5-158
:SENSe{1-16}:CORRection:COLLect:PORT{12}:TFRF 5-158
:SENSe{1-16}:CORRection:COLLect:PORT{12}:TFRR 5-158
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:FREQuency <NRf> 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:FREQuency? 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LABEL <string> 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LABEL? 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LENGth <NRf> 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LENGth? 5-159
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LOSS <NRf> 5-160
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:LOSS? 5-160
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:RECIProcal <char> 5-160
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:RECIProcal? 5-160
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:SERial <string> 5-160
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:SERial? 5-160
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:Z0 <NRf> 5-161
:SENSe{1-16}:CORRection:COLLect:PORT{12}:THRu:Z0? 5-161
:SENSe{1-16}:CORRection:COLLect:REFerence:Z0 <NRf> 5-188
:SENSe{1-16}:CORRection:COLLect:REFerence:Z0? 5-188
:SENSe{1-16}:CORRection:COLLect:SAVe 5-188
:SENSe{1-16}:CORRection:COLLect:TFR:CLEar 5-188
:SENSe{1-16}:CORRection:COLLect:THRu:ADD <char> 5-196
:SENSe{1-16}:CORRection:COLLect:THRu:CLEar 5-196
:SENSe{1-16}:CORRection:COLLect:WAVeguide:DIELectric <NRf> 5-199
:SENSe{1-16}:CORRection:COLLect:WAVeguide:DIELectric? 5-199
:SENSe{1-16}:CORRection:COLLect:WAVeguide:FREQuency <NRf> 5-199
:SENSe{1-16}:CORRection:COLLect:WAVeguide:FREQuency? 5-199
:SENSe{1-16}:CORRection:COLLect:WAVeguide:KIT <char> 5-200
:SENSe{1-16}:CORRection:COLLect:WAVeguide:KIT? 5-200
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LABel <string> 5-200
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LABel? 5-200
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:L0 <NRf> 5-201
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:L0? 5-201
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:OFFSet <NRf> 5-201
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:OFFSet? 5-201
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:R <NRf> 5-201
:SENSe{1-16}:CORRection:COLLect:WAVeguide:LOAD:R? 5-201
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C0 <NRf> 5-202
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C0? 5-202
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C1 <NRf> 5-202
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C1? 5-202
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C2 <NRf> 5-202

:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C2? 5-202
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C3 <NRf> 5-203
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:C3? 5-203
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:OFFSet <NRf> 5-203
:SENSe{1-16}:CORRection:COLLect:WAVeguide:OPEN:OFFSet? 5-203
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SERial <string> 5-203
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SERial? 5-203
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT1:OFFSet <NRf> 5-204
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT1:OFFSet? 5-204
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT2:OFFSet <NRf> 5-204
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT2:OFFSet? 5-204
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT3:OFFSet <NRf> 5-204
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SHORT3:OFFSet? 5-204
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SLOAD:MINF <NRf> 5-205
:SENSe{1-16}:CORRection:COLLect:WAVeguide:SLOAD:MINF? 5-205
:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:1P2PF 5-189
:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:1P2PR 5-189
:SENSe{1-16}:CORRection:COLLect[:CALa]:PORT{12}:FULL2 5-190
:SENSe{1-16}:CORRection:COLLect[:METHod]:1P2PF 5-192
:SENSe{1-16}:CORRection:COLLect[:METHod]:1P2PR 5-192
:SENSe{1-16}:CORRection:COLLect[:METHod]:FULL1 5-192
:SENSe{1-16}:CORRection:COLLect[:METHod]:LINE <char> 5-193
:SENSe{1-16}:CORRection:COLLect[:METHod]:LINE? 5-193
:SENSe{1-16}:CORRection:COLLect[:METHod]:LOAD <char> 5-194
:SENSe{1-16}:CORRection:COLLect[:METHod]:LOAD? 5-194
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT <char> 5-194
:SENSe{1-16}:CORRection:COLLect[:METHod]:PORT? 5-194
:SENSe{1-16}:CORRection:COLLect[:METHod]:REFTHru 5-195
:SENSe{1-16}:CORRection:COLLect[:METHod]:RESP1 5-194
:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRB 5-195
:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRF 5-195
:SENSe{1-16}:CORRection:COLLect[:METHod]:TFRR 5-196
:SENSe{1-16}:CORRection:COLLect[:METHod]:TYPe? 5-196
:SENSe{1-16}:CORRection:EXTension:PORT{1-2} <NRf> 5-206
:SENSe{1-16}:CORRection:EXTension:PORT{1-2}? 5-206
:SENSe{1-16}:CORRection:ISOLation:STATe <char> 5-207
:SENSe{1-16}:CORRection:ISOLation:STATe? 5-207
:SENSe{1-16}:CORRection:STATe <char> 5-208
:SENSe{1-16}:CORRection:STATe? 5-208
:SENSe{1-16}:FREQuency:CENTer <NRf> 5-209
:SENSe{1-16}:FREQuency:CENTer? 5-209
:SENSe{1-16}:FREQuency:CW <NRf> 5-209
:SENSe{1-16}:FREQuency:CW? 5-209
:SENSe{1-16}:FREQuency:DATA <block> 5-210
:SENSe{1-16}:FREQuency:DATA? 5-210
:SENSe{1-16}:FREQuency:SPAN <NRf> 5-210
:SENSe{1-16}:FREQuency:SPAN? 5-210
:SENSe{1-16}:FREQuency:STARt <NRf> 5-210
:SENSe{1-16}:FREQuency:STARt? 5-210
:SENSe{1-16}:FREQuency:STOP <NRf> 5-210
:SENSe{1-16}:FREQuency:STOP? 5-210
:SENSe{1-16}:FSEGMent:ADD 5-211
:SENSe{1-16}:FSEGMent:CLEar 5-211
:SENSe{1-16}:FSEGMent:COUNt? 5-212
:SENSe{1-16}:FSEGMent:CWMODe[:STATe] <char> 5-212
:SENSe{1-16}:FSEGMent:CWMODe[:STATe]? 5-212
:SENSe{1-16}:FSEGMent:DATA? 5-212
:SENSe{1-16}:FSEGMent:DISPlay <char> 5-212
:SENSe{1-16}:FSEGMent:DISPlay? 5-212
:SENSe{1-16}:FSEGMent:FREQuency:ACTive:STARt? 5-213
:SENSe{1-16}:FSEGMent:FREQuency:ACTive:STOP? 5-213

:SENSe{1-16}:FSEGMent:FREQuency:FSTEp <NRf> 5-213
:SENSe{1-16}:FSEGMent:FREQuency:FSTEp? 5-213
:SENSe{1-16}:FSEGMent:FREQuency:FSTOp <NRf> 5-214
:SENSe{1-16}:FSEGMent:FREQuency:FSTOp? 5-214
:SENSe{1-16}:FSEGMent:FREQuency:STARt <NRf> 5-214
:SENSe{1-16}:FSEGMent:FREQuency:STARt? 5-214
:SENSe{1-16}:FSEGMent:FREQuency:STOP <NRf> 5-214
:SENSe{1-16}:FSEGMent:FREQuency:STOP? 5-214
:SENSe{1-16}:FSEGMent:FREQuency[:CW][:FIXed] <NRf> 5-215
:SENSe{1-16}:FSEGMent:FREQuency[:CW][:FIXed]? 5-215
:SENSe{1-16}:FSEGMent:MAXPoints? 5-215
:SENSe{1-16}:FSEGMent:SPAntype <char> 5-215
:SENSe{1-16}:FSEGMent:SPAntype? 5-215
:SENSe{1-16}:FSEGMent:SWEep:MAXimize 5-216
:SENSe{1-16}:FSEGMent:SWEep:POINt <NR1> 5-216
:SENSe{1-16}:FSEGMent:SWEep:POINt? 5-216
:SENSe{1-16}:FSEGMent[:STATe] <char> 5-217
:SENSe{1-16}:FSEGMent[:STATe]? 5-217
:SENSe{1-16}:FSEGMent{1-50}:CWMODe[:STATe] <char> 5-218
:SENSe{1-16}:FSEGMent{1-50}:CWMODe[:STATe]? 5-218
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTArt <NRf> 5-218
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTArt? 5-218
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTEp <NRf> 5-219
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTEp? 5-219
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTOp <NRf> 5-219
:SENSe{1-16}:FSEGMent{1-50}:FREQuency:FSTOp? 5-219
:SENSe{1-16}:FSEGMent{1-50}:FREQuency[:CW][:FIXed] <NRf> 5-219
:SENSe{1-16}:FSEGMent{1-50}:FREQuency[:CW][:FIXed]? 5-219
:SENSe{1-16}:FSEGMent{1-50}:SPAntype <char> 5-220
:SENSe{1-16}:FSEGMent{1-50}:SPAntype? 5-220
:SENSe{1-16}:FSEGMent{1-50}:SWEep:POINt <NR1> 5-220
:SENSe{1-16}:FSEGMent{1-50}:SWEep:POINt? 5-220
:SENSe{1-16}:FSEGMent{1-50}[:STATe] <char> 5-221
:SENSe{1-16}:FSEGMent{1-50}[:STATe]? 5-221
:SENSe{1-16}:HOLD:FUNCtion <char> 5-222
:SENSe{1-16}:HOLD:FUNCtion? 5-222
:SENSe{1-16}:ISEGMent:ADD 5-225
:SENSe{1-16}:ISEGMent:CLEar 5-225
:SENSe{1-16}:ISEGMent:COUNt? 5-225
:SENSe{1-16}:ISEGMent:CWMODe[:STATe] <char> 5-226
:SENSe{1-16}:ISEGMent:CWMODe[:STATe]? 5-226
:SENSe{1-16}:ISEGMent:DATA? 5-226
:SENSe{1-16}:ISEGMent:FREQuency:FSTArt <NRf> 5-226
:SENSe{1-16}:ISEGMent:FREQuency:FSTArt? 5-226
:SENSe{1-16}:ISEGMent:FREQuency:FSTEp <NRf> 5-226
:SENSe{1-16}:ISEGMent:FREQuency:FSTEp? 5-226
:SENSe{1-16}:ISEGMent:FREQuency:FSTOp <NRf> 5-227
:SENSe{1-16}:ISEGMent:FREQuency:FSTOp? 5-227
:SENSe{1-16}:ISEGMent:FREQuency[:CW][:FIXed] <NRf> 5-227
:SENSe{1-16}:ISEGMent:FREQuency[:CW][:FIXed]? 5-227
:SENSe{1-16}:ISEGMent:INDex:ACTive:STARt? 5-228
:SENSe{1-16}:ISEGMent:INDex:ACTive:STOP? 5-228
:SENSe{1-16}:ISEGMent:INDex:STARt <NRf> 5-228
:SENSe{1-16}:ISEGMent:INDex:STARt? 5-228
:SENSe{1-16}:ISEGMent:INDex:STOP <NRf> 5-228
:SENSe{1-16}:ISEGMent:INDex:STOP? 5-228
:SENSe{1-16}:ISEGMent:MAXPoints? 5-229
:SENSe{1-16}:ISEGMent:POINt? 5-229
:SENSe{1-16}:ISEGMent:SPAntype <char> 5-229
:SENSe{1-16}:ISEGMent:SPAntype? 5-229
:SENSe{1-16}:ISEGMent:SWEep:MAXimize 5-230

:SENSe{1-16}:ISEGMent:SWEep:POINt <NR1> 5-230
:SENSe{1-16}:ISEGMent:SWEep:POINt? 5-230
:SENSe{1-16}:ISEGMent[:STATe] <char> 5-230
:SENSe{1-16}:ISEGMent[:STATe]? 5-230
:SENSe{1-16}:ISEGMent{1-50}:CWMODe[:STATe] <char> 5-231
:SENSe{1-16}:ISEGMent{1-50}:CWMODe[:STATe]? 5-231
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTArt <NRf> 5-231
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTArt? 5-231
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTEp <NRf> 5-232
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTEp? 5-232
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTOp <NRf> 5-232
:SENSe{1-16}:ISEGMent{1-50}:FREQuency:FSTOp? 5-232
:SENSe{1-16}:ISEGMent{1-50}:FREQuency[:CW][:FIXed] <NRf> 5-232
:SENSe{1-16}:ISEGMent{1-50}:FREQuency[:CW][:FIXed]? 5-232
:SENSe{1-16}:ISEGMent{1-50}:SPAntype <char> 5-233
:SENSe{1-16}:ISEGMent{1-50}:SPAntype? 5-233
:SENSe{1-16}:ISEGMent{1-50}:SWEep:POINt <NR1> 5-233
:SENSe{1-16}:ISEGMent{1-50}:SWEep:POINt? 5-233
:SENSe{1-16}:ISEGMent{1-50}[:STATe] <char> 5-234
:SENSe{1-16}:ISEGMent{1-50}[:STATe]? 5-234
:SENSe{1-16}:RECEiver:CONFiguration <char> 5-235
:SENSe{1-16}:RECEiver:CONFiguration? 5-235
:SENSe{1-16}:SEGMent:TYPe? 5-235
:SENSe{1-16}:SWEep:CW:POINt <NRf> 5-236
:SENSe{1-16}:SWEep:CW:POINt? 5-236
:SENSe{1-16}:SWEep:CW[:STATe] <char> 5-236
:SENSe{1-16}:SWEep:CW[:STATe]? 5-236
:SENSe{1-16}:SWEep:DELay <NRf> 5-236
:SENSe{1-16}:SWEep:DELay? 5-236
:SENSe{1-16}:SWEep:POINt <NRf> 5-237
:SENSe{1-16}:SWEep:POINt? 5-237
:SENSe{1-16}:SWEep:TYPe <char> 5-237
:SENSe{1-16}:SWEep:TYPe? 5-237
:SOURce{1-16}:POWer <char> 5-238
:STATus:OPERation:CONDition? 5-239
:STATus:OPERation:ENABle <NRf> 5-239
:STATus:OPERation:ENABle? 5-239
:STATus:OPERation:NTRansition <NRf> 5-239
:STATus:OPERation:NTRansition? 5-239
:STATus:OPERation:PTRansition <NRf> 5-239
:STATus:OPERation:PTRansition? 5-239
:STATus:OPERation[:EVENt]? 5-240
:STATus:QUEStionable:CONDition? 5-241
:STATus:QUEStionable:ENABle <NRf> 5-241
:STATus:QUEStionable:ENABle? 5-241
:STATus:QUEStionable:LIMit:CONDition? 5-241
:STATus:QUEStionable:LIMit:ENABle <NRf> 5-241
:STATus:QUEStionable:LIMit:ENABle? 5-241
:STATus:QUEStionable:LIMit:NTRansition <NRf> 5-242
:STATus:QUEStionable:LIMit:NTRansition? 5-242
:STATus:QUEStionable:LIMit:PTRansition <NRf> 5-242
:STATus:QUEStionable:LIMit:PTRansition? 5-242
:STATus:QUEStionable:LIMit[:EVENt]? 5-242
:STATus:QUEStionable:NTRansition <NRf> 5-242
:STATus:QUEStionable:NTRansition? 5-242
:STATus:QUEStionable:PTRansition <NRf> 5-243
:STATus:QUEStionable:PTRansition? 5-243
:STATus:QUEStionable[:EVENt]? 5-243
:SYSTem:COMMunicate:TCPIP:ADDRess? 5-244
:SYSTem:COMMunicate:TCPIP:GATE? 5-244
:SYSTem:COMMunicate:TCPIP:HDW? 5-244

:SYSTem:COMMunicate:TCPIP:MASK? 5-244
:SYSTem:COMMunicate:TCPIP:PORT <NRf> 5-245
:SYSTem:COMMunicate:TCPIP:PORT? 5-245
:SYSTem:ERRor:CLEar 5-245
:SYSTem:ERRor:COUNt? 5-245
:SYSTem:ERRor:QUEue? 5-245
:SYSTem:ERRor[:NEXt]? 5-245
:SYSTem:FACTorycal:DELete 5-246
:SYSTem:FACTorycal:RECover 5-246
:SYSTem:HOLD:RF[:STATe] <char> 5-246
:SYSTem:HOLD:RF[:STATe]? 5-246
:SYSTem:IFCalibration:TRIGger 5-246
:SYSTem:POINt:MAXimum? 5-246
:SYSTem:PORT:COUNt? 5-247
:SYSTem:POWerup:FILe <string> 5-247
:SYSTem:POWerup:FILe? 5-247
:SYSTem:POWerup:TYPe <char> 5-247
:SYSTem:POWerup:TYPe? 5-247
:SYSTem:PRESet 5-247
:SYSTem:PRESet:FILe <string> 5-248
:SYSTem:PRESet:FILe? 5-248
:SYSTem:PRESet:TYPe <char> 5-249
:SYSTem:PRESet:TYPe? 5-249
:SYSTem:PRESet:ZERo 5-249
:TRIGger[:SEQuence]:SOURce <char> 5-250
:TRIGger[:SEQuence]:SOURce? 5-250
:TRIGger[:SEQuence][:IMMediate][:REMote] 5-250
:TRIGger[:SEQuence][:REMote]:SINGle 5-252
*CLS 3-2
*DDT <Arbitrary Block> | <String> 3-2
*DDT? 3-2
*ESE <NRf> 3-3
*ESE? 3-3
*ESR? 3-3
*IDN? 3-3
*OPC 3-4
*OPC? 3-6
*OPT? 3-6
*RST 3-7
*SRE <NRf> 3-8
*SRE? 3-8
*STB? 3-8
*TRG 3-9
*TST? 3-9
*WAI 3-9

Anritsu

10410-00338

U

Anritsu utilizes recycled paper and environmentally conscious inks and toner.

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