Agilent N9322C Spectrum Analyzer

Programmer's Guide

Agilent Technologies

Notices

Part Number

N9322-90023

Edition

First Edition, October 2012

Agilent Technologies, Inc.
No. 116 Tuo Xin West 1st Street
Hi-Tech Industrial Zone (South)
Chengdu 610041, China

Software Revision

This guide is valid for Version A.04.20 or later of the N9322C spectrum analyzer firmware.

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Restricted Rights Legend

U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

In This Guide...

This guide contains programming information for the N9322C Spectrum Analyzer.

1 Getting Started

Prepare for the remote control.

2 Programming Fundamentals

A quick overview of the SCPI programming.

3 Status Registers

Introduction of the status registers.

4 Programming Example

How to accomplish the basic applications in programming.

5 Command Reference

Describe every programming command ant the related softkeys' functions in detail.

For more information about the N9322C handheld spectrum analyzer, please refer to:

N9322C spectrum analyzer:

www.agilent.com/find/n9322c

Contents

Contents

1 Getting Started

The purpose of this chapter is to serve as a reminder of SCPI (Standard Commands for Programmable Instruments) fundamentals to those who have previous experience in programming SCPI. This chapter is not intended to teach you everything about the SCPI programming language. If you are using an optional programming compatibility modes, you should refer to the manual that came with the option.

Remotely Operating the Analyzer

The analyzer provides both the USB and LAN connection which allows you to set up a remote operation environment with a controller computer. A controller computer could be a personal computer (PC), a minicomputer. Some intelligent instruments also function as controllers.

Computer Requirement for Remote Operation

Usually, you need to prepare an compatible PC with the following requirements to set up a remote operation environment:

Processor: 450 MHz Pentium® II or higher required

Operating system: Microsoft® Windows® XP or Home Editon, Service Pack 1 or later; Windows® 2000 Professional, service pack 4 or later

Available memory: 128 MB or higher required

Available disk space: 175 MB or greater required

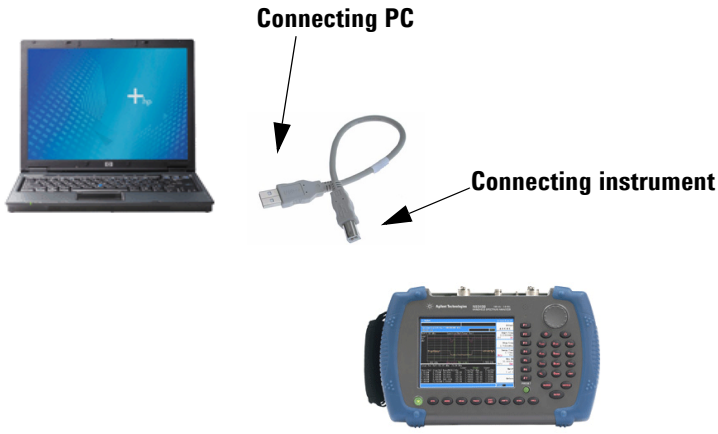
Connecting the Analyzer via the USB Port

No extra driver is required to connect the analyzer via the USB port to a PC. All you need is the Agilent IO libraries suite and you can find this IO libraries suite in the documentation CD in the shipment along with your analyzer. Or download the IO libraries suite from Agilent website:

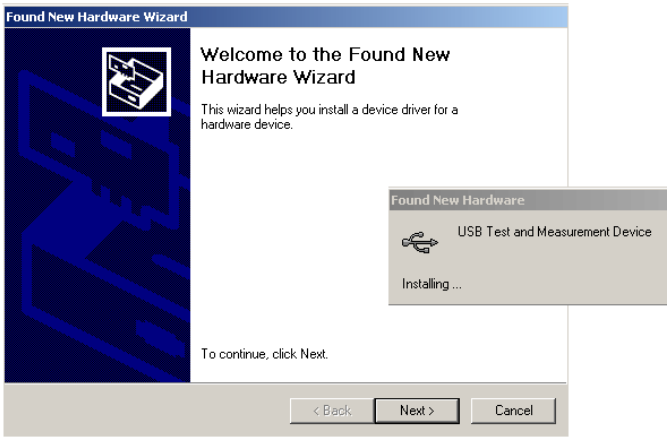
http://www.agilent.com/find/iolib

Refer to the following steps to finish the connection:

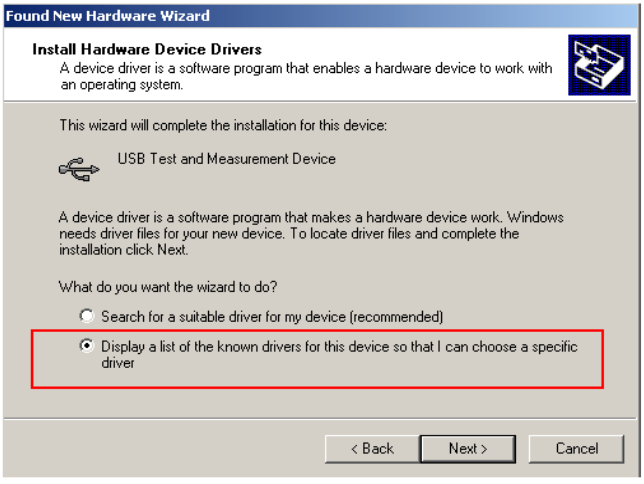
1. Install Agilent IO libraries suite on your PC
2. Switch on the analyzer
3. Connect the analyzer to a PC with a USB cable.



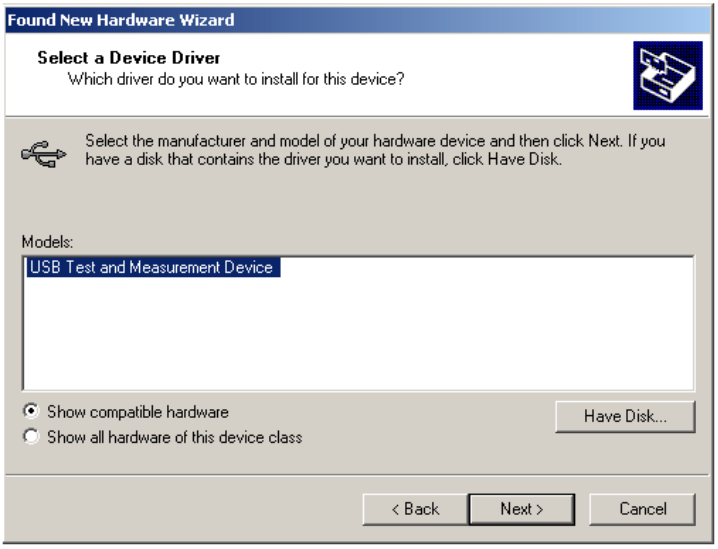
4 After a while, the PC finds your analyzer as a new hardware and prompts a message saying “Found new hardware...”. A **Found New Hardware Wizard** is initiated immediately.



5 Select **Display a list...**



7 PC will detect the instrument automatically. The item **"USB Test and Measurement Device"** displays in the pop-up window. Select it and press **Next**.



The wizard will guide you through the rest of installation till the driver is installed. Run Agilent IO libraries suite, the analyzer will be detected automatically.

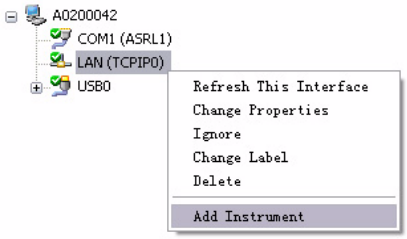
Connecting the Analyzer via the LAN Port

No extra driver is required to connect the analyzer via the LAN port to a PC. All you need is the Agilent IO libraries suite in the Product CD *Help Kit*. Or refer to the link below to download the IO libraries suite:

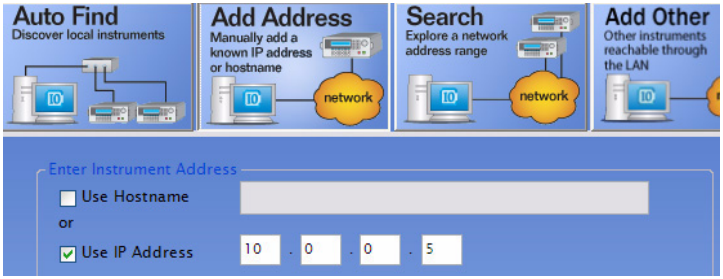
http://www.agilent.com/find/iolib

Please refer to the following steps to finish the connection:

1. Switch on the analyzer.
2. Connect the spectrum analyzer to a PC with a LAN cable.
3. Press **[SYS] > {Setting} > {IP Admin} > {IP address}** to set IP address for the instrument. For example, set "10.0.0.5" as the IP address for the instrument. Press **{Apply}** as
4. Run Agilent Connection Expert in IO libraries suite. Right-click on the LAN (TCPIP0) icon, select "Add Instrument' in the pop-up menu. The "Add LAN Instruments" window displays for the IP configuration.

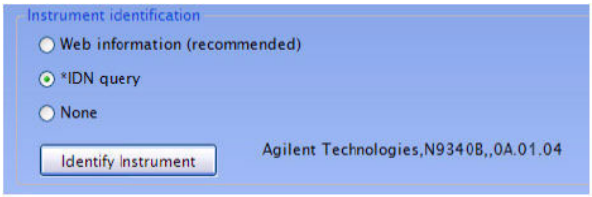


5. Select "Add Address", check "Use IP Address" in the window and input the IP address as the instrument IP address you set before.



6 Press “Test Connection” to check the LAN connection. The figure below indicates that the connection is ready.

7 Check “*IDN query” and press “Identify Instrument”. The instrument information shows the firmware revision and product number. The analyzer is ready for your further programming.



1 Getting Started

2
Programming Fundamentals

The purpose of this chapter is to serve as a reminder of SCPI (Standard Commands for Programmable Instruments) fundamentals to those who have previous experience in programming SCPI. This chapter is not intended to teach you everything about the SCPI programming language. If you are using an optional programming compatibility modes, you should refer to the manual that came with the option.

Overview

This section is not intended to teach you everything about the SCPI (Standard Commands for Programmable Instruments) programming language. The SCPI Consortium or IEEE provides that level of detailed information.

Programming with SCPI requires knowledge of:

- Computer programming languages, such as C, C++, and Microsoft®Visual Basic®.
- The language of your instrument. The analyzer employs SCPI as its programming language.

The semantic requirements of your controller's language determine how the programming commands and responses are handled in your application program.

SCPI Language Basics

SCPI is an ASCII-based instrument command language designed for test and measurement instruments, with the goal of reducing automatic test equipment (ATE) program development time.

SCPI accomplishes this goal by providing a consistent programming environment for instrument control and data usage. This consistent programming environment is achieved by the use of defined program messages, instrument responses, and data formats across all SCPI instruments.

By providing a consistent programming environment, replacing one SCPI instrument with another SCPI instrument in a system will usually require less effort than with non-SCPI instrument.

SCPI is not a standard which completely provides for interchangeable instrumentation. SCPI helps move toward interchangeability by defining instrument commands and responses, but not functionality, accuracy, resolution, etc.

Common Terms used in this Book

Terms	Description
Controller	Any computer used to communicate with an instrument. A controller can be a personal computer (PC), a minicomputer, or a plug-in card in a card cage. Some intelligent instruments can also function as controllers.
Instrument	Any device that implements SCPI. Most instruments are electronic measurement or stimulus devices, but this is not a requirement. Similarly, most instruments use a GPIB or RS-232 or USB interface for communication. The same concepts apply regardless of the instrument function or the type of interface used.
Command	An instruction. You combine commands to form messages that control instruments to complete a specified task. In general, a command consists of mnemonics (keywords), parameters and punctuation.
Query	A special type of command. Queries instruct the instrument to make response data available to the controller. Query keywords always end with a question mark, ? .

The SCPI Consortium or IEEE can provide detailed information on the subject of SCPI programming. Refer to IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation.* New York, NY, 1987, or to IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

Command Categories

The SCPI command falls into two categories:

- Subsystem commands that simulate front panel keystrokes
- Common commands that are unique and have no front panel equivalent

Use a computer to control the instrument (but operate the power/standby switch manually). Computer programming procedures for the instrument involve selecting a programming statement and then adding the specified programming codes to that statement to achieve the desired operating conditions.

For more specific command instructions, please refer to Chapter 5, “Command Reference,” starting on page 49.

Command Syntax

A command consists of mnemonics (keywords), parameters and punctuation. Before you start to program your signal generator, familiarize yourself with the standard notation of each of them.

Command Mnemonics (keywords)	Many commands have both a long and a short form: use either one. (a combination of the two is not allowed). Consider the :FREQuency command for example: • Short form :FREQ • Long form :FREQUENCY SCPI is not case sensitive, so fREquEncy is just as valid as FREQUENCY, but FREQ and FREQUENCY are the only valid forms of the FREQuency command. In this documentation, upper case letters indicate the short form of the keyword. The lower case letters indicate the long form of the keyword.
Punctuation	• A vertical bar "|" dictates a choice of one element from a list. For example: <A>|<B> indicates that either A or B can be selected, but not both. • Square brackets "[ ]" indicates that the enclosed items are optional. • Angle brackets "< >" indicates a variable items to be entered to represent user choices. • A question mark "?" after a subsystem command indicates that the command is a query. The returned information, <value> varies in format according to the type of the field.
Separator	• A colon ":" seperates keywords of different levels. The colon before the root keyword is usually omitted. • A space separates a keyword and a parameter, as well as a parameter and a unit.

Command Statement Rules Overview

Besides the standard notation of SCPI described above, please remember the following rules in programming:

- command statements read from left to right
- use either long form or short form of keywords, but do not use both
- no separating space between the keywords, only use a colon to separate keywords of different levels
- always separating a keyword from a variable with a space
- always separating a variable from its unit with a space (if variable has a unit).

Command Example

A typical command is made up of key words set off by colons. The key words are followed by parameters that can be followed by optional units.

Example 1 `:TRIGger:SEQuence:VIDeo:LEVel 2.5V`

The instrument does not distinguish between upper and lower case letters. In the documentation, upper case letters indicate the short form of the key word. The upper and lower case letters, together, indicate the long form of the key word. Either form may be used in the command.

Example 2 `:Trig:Seq:Vid:Lev 2.5V` is the same as `:trigger:sequence:video:level 2.5V`.

NOTE The command `:TRIGG:Sequence:Video:Level 2.5V` is not valid because `:TRIGG` is neither the long, nor the short form of the command.

Creating Valid Commands

Commands are not case sensitive and there are often many different ways of writing a particular command. These are examples of valid commands for a given command syntax:

Command Syntax	Sample Valid Commands

`[:SENSe]:BANDwidth[:RESolution]<freq>`	The following sample commands are all identical. They will all cause the same result.
	`:Sense:Band:Res 1700`
	`:BANDWIDTH:RESOLUTION 1.7e3`
	`:sens:band 1.7KHZ`
	`:SENS:band 1.7E3Hz`
	`:band 1.7kHz`
	`:bandwidth:RES 1.7e3Hz`

`:CALCulate:MARKer[1]|2|3|4:Y?`	The last command below returns different results than the commands above it. The number 3 in the command causes this. See the command description for more information.
	`:CALC:MARK:Y?`
	`:calc:mark:y?`
	`:CALC:MARK2:Y?`

`[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle`	`DET:FUNC NEG` `:Sense:Detector:Function Sample`

`:INITiate:CONTinuous OFF|ON|0|1`	The sample commands below are identical.
	`:INIT:CONT ON`
	`:init:continuous 1`

Program and Response Messages

To understand how your instrument and controller communicate using SCPI, you must understand the concepts of program and response messages.

Program Messages

Program messages are the formatted data sent from the controller to the instrument. Conversely, response messages are formatted data sent from the instrument to the controller. Program messages contain one or more commands, and response messages contain one or more responses.

Response Messages

The controller may send commands at any time, but the instrument sends responses only when query commands is received. All query mnemonics end with a question mark. Queries return either measured values or internal instrument settings.

Forgiving Listening and Precise Talking

SCPI uses the concept of forgiving listening and precise talking outlined in IEEE 488.2.

Forgiving listening means that instruments are very flexible in accepting various command and parameter formats. For example, the spectrum analyzer accepts either `:FREQuency:CENTer:STEP:AUTO ON` or `:FREQuency:CENTer:STEP:AUTO 1`

Precise talking means that the response format for a particular query is always the same. For example, if you query RF output state when it is on (using `:FREQuency:CENTer:STEP:AUTO?`), the response is always 1, regardless of if you previously sent `:FREQuency:CENTer:STEP:AUTO ON` or `:FREQuency:CENTer:STEP:AUTO 1`.

Parameters in Commands

There are four basic types of parameters: boolean, key words, variables and arbitrary block program data.

Boolean

The expression `OFF|ON|0|1` is a two state boolean-type parameter. The numeric value 0 is equivalent to OFF. Any numeric value other than 0 is equivalent to ON. The numeric values of 0 or 1 are commonly used in the command instead of OFF or ON, and queries of the parameter always return a numeric value of 0 or 1.

Key Word

The parameter key words that are allowed for a particular command are defined in the command description and are separated with a vertical slash.

Units

Numerical variables may include units. The valid units for a command depends on the variable type being used. See the following variable descriptions. If no units are sent, the indicated default units will be used. Units can follow the numerical value with, or without, a space.

Variable

A variable can be entered in exponential format as well as standard numeric format. The appropriate variable range and its optional units are defined in the command description.

Variable Parameters

<ampl>, <rel_ampl> The <ampl> (amplitude) parameter and the <rel_ampl> (relative amplitude) parameter consist of a rational number followed by optional units. Acceptable units for <ampl> include: V, mV, V, dBm, dBmV, dBuV, Watts, W. <rel_ampl> units are given in dB.

<file_name> A file name parameter is the name of your file, is not used in the SCPI command string.

<freq> A frequency parameter is a positive rational number followed by optional units. The default unit is Hz. Acceptable units include: Hz, kHz, MHz, GHz.

<integer> There are no units associated with an integer parameter.

<number> A number parameter is a member of the set of positive or negative intriguers and including zero. Fractional numbers are included in the number parameter. There are no units associated with a number parameter.

<percent> A percent parameter is a rational number between 0 and 100, with no units.

<rel_power> A relative power parameter is a positive rational number followed by optional units. The default units are dB. Acceptable units are dB only.

<string> A string parameter includes a series of alpha numeric characters.

<time> A time parameter is a rational number followed by optional units. The default units are seconds. Acceptable units include: S, MS, US.

3
Status Registers

This chapter contains a comprehensive description of status registers explaining what status registers are and how to use them so you can use a program to monitor the instrument. Information about all of the bits of the status registers is also provided.

Overview

When you are programming the instrument you may need to monitor instrument status to check for error conditions or monitor changes. You need to determine the state of certain instrument events/conditions by programming the status register system.

IEEE common commands (those beginning with *) access the higher-level summary registers. To access the information from specific registers you would use the STATus commands. The STATus subsystem remote commands set and query the status hardware registers. This system of registers monitors various events and conditions in the instrument. Software written to control the instrument may need to monitor some of these events and conditions.

What are Status Registers

The status system contains multiple registers that are arranged in a hierarchical order. The lower-level status registers propagate their data to the higher-level registers in the data structures by means of summary bits. The status byte register is at the top of the hierarchy and contains general status information for the instrument's events and conditions. All other individual registers are used to determine the specific events or conditions.

Each register set is made up of five registers:

Condition Register It reports the real-time state of the signals monitored by this register set. There is no latching or buffering for a condition register.

Positive Transition Register This filter register controls which signals will set a bit in the event register when the signal makes a low to high transition (when the condition bit changes from 0 to 1).

Negative Transition Register This filter register controls which signals will set a bit in the event register when the signal makes a high to low transition (when the condition bit changes from 1 to 0).

Event Register It latches any signal state changes, in the way specified by the filter registers. Bits in the event register are never cleared by signal state changes. Event registers are cleared when read. They are also cleared by *CLS and by presetting the instrument.

Event Enable Register It controls which of the bits, being set in the event register, will be summarized as a single output for the register set. Summary bits are then used by the next higher register.

Access the status registers

There are two different methods to access the status registers:

- Common Commands Accesses and Controls
- Status Subsystem Commands

What are Status Register SCPI Commands

Most monitoring of the instrument conditions is done at the highest level using the IEEE common commands indicated below. Complete command descriptions are available in the IEEE commands section at the beginning of the language reference. Individual status registers can be set and queried using the commands in the STATus subsystem of the language reference.

- ***CLS** (clear status) clears the status byte by emptying the error queue and clearing all the event registers.
- ***ESE, *ESE?** (event status enable) sets and queries the bits in the enable register part of the standard event status register.
- ***ESR?** (event status register) queries and clears the event register part of the standard event status register.
- ***SRE,*SRE?** (service request enable) sets and queries the value of the service request enable register.
- ***STB?** (status byte) queries the value of the status byte register without erasing its contents.

How to use the Status Registers

A program often needs to detect and manage error conditions or changes in instrument status. The polling method for you to programmatically access the information in status registers.

In the polling method, the instrument has a passive role. It only tells the controller that conditions have changed when the controller asks the right question. In the SRQ method, the instrument takes a more active role. It tells the controller when there has been a condition change without the controller asking. Either method allows you to monitor one or more conditions.

The polling method works well if you do not need to know about changes the moment they occur. To detect a change using the polling method, the program must repeatedly read the registers.

To monitor a condition:

— Determine which register contains the bit that reports the condition.

— Send the unique SCPI query that reads that register.

— Examine the bit to see if the condition has changed.

You can monitor conditions in different ways.

- Check the instrument hardware and firmware status.

Do this by querying the condition registers which continuously monitor status. These registers represent the current state of the instrument. Bits in a condition register are updated in real time. When the condition monitored by a particular bit becomes true, the bit is set to 1. When the condition becomes false, the bit is reset to 0.

- Monitor a particular condition (bit).

You can enable a particular bit(s), using the event enable register. The instrument will then monitor that particular condition(s). If the bit becomes true (0 to 1 transition) in the event register, it will stay set until the event register is cleared. Querying the event register allows you to detect that this condition occurred even if the condition no longer exists. The event register can only be cleared by querying it or sending the `*CLS` command.

- Monitor a particular type of change in a condition (bit).

— The transition registers are preset to register if the condition goes from 0 to 1 (false to true, or a positive transition).

— This can be changed so the selected condition is detected if the bit goes from 1 to 0 (true to false, or a negative transition).

— It can also be set for both types of transitions occurring.

— Or it can be set for neither transition. If both transition registers are set to 0 for a particular bit position, that bit will not be set in the event register for either type of change.

Status Register Examples

Each bit in a register is represented by a numerical value based on its location. See figure below. This number is sent with the command to enable a particular bit. If you want to enable more than one bit, you would send the sum of all the bits that you want to monitor.

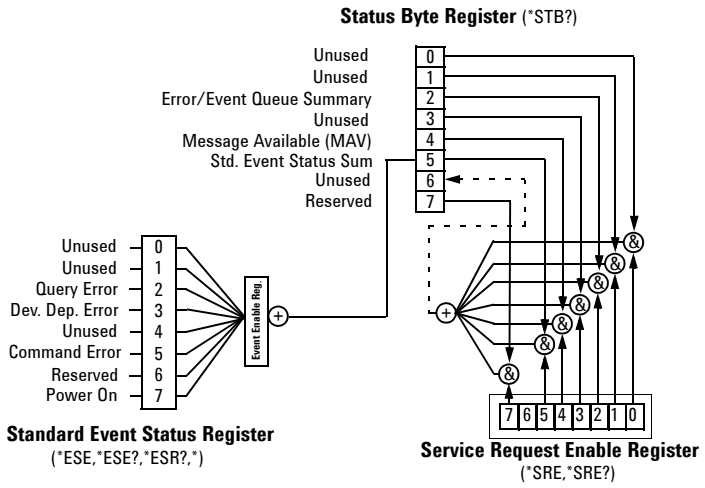
Example

1. To enable bit 0 and bit 6 of standard event status register, you would send the command `*ESE 65` because 1 + 64 = 65.
2. The results of a query are evaluated in a similar way. If the `*STB?` command returns a decimal value of 140, (140 = 128 + 8 + 4) then bit 7 is true, bit 3 is true and bit 2 is true.

Status Register System

The hardware status registers are combined to form the instrument status system. Specific status bits are assigned to monitor various aspects of the instrument operation and status. See the following diagram of the status system for information about the bit assignments and status register interconnections.

Figure 1 Agilent N9322C Status Register System



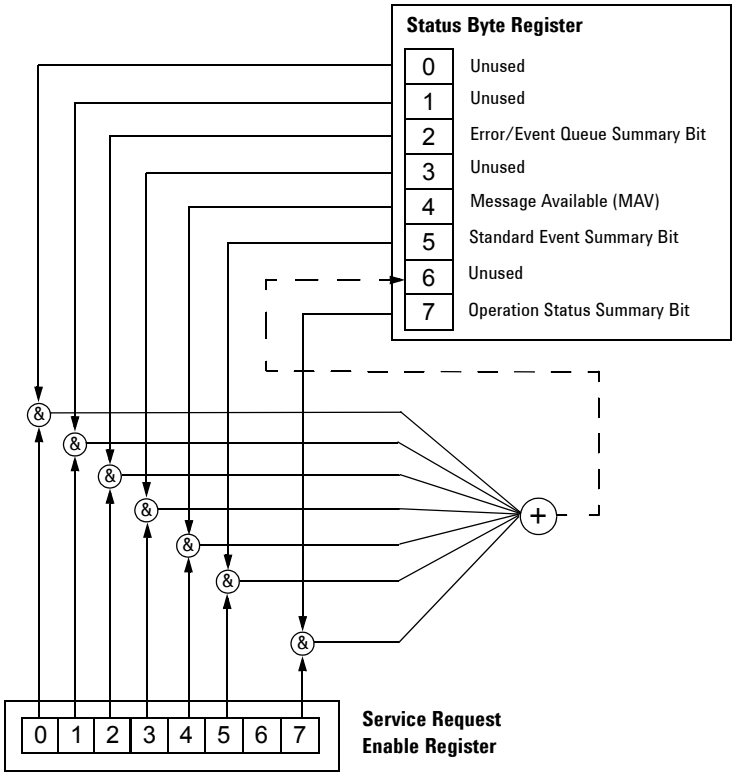
Setting and Querying the Status Register

Each bit in a register is represented by a numerical value based on its location. This number is sent with the command to enable a particular bit. To enable more than one bit, send the sum of all of the bits involved.

For example, to enable bit 0 and bit 6 of the standard event status register, you would send the command `*ESE 65 (1 + 64)`.

The results of a query are evaluated in a similar way. If the **`*STB?`** command returns a decimal value of 140, (140 = 128 + 8 + 4) then bit 7 is true, bit 3 is true, and bit 2 is true.

The Status Byte Register



The RQS bit is read and reset by a serial poll. The same bit position (MSS) is read, non-destructively by the *STB? command. If you serial poll bit 6 it is read as RQS, but if you send *STB it reads bit 6 as MSS. For more information refer to IEEE 488.2 standards, section 11.

The status byte register contains the following bits:

Bit	Description
0,1	`Unused:` These bits are always set to 0.
2	`Error/Event Queue Summary Bit:` A 1 in this bit position indicates that the SCPI error queue is not empty. The SCPI error queue contains at least one error message.
3	`Questionable Status Summary Bit:` A 1 in this bit position indicates that the questionable status summary bit has been set. The questionable status event register can then be read to determine the specific condition that caused this bit to be set.
4	`Message Available (MAV):` A 1 in this bit position indicates that the analyzer has data ready in the output queue. There are no lower status groups that provide input to this bit.
5	`Standard Event Status Summary Bit:` A 1 in this bit position indicates 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 this bit to be set.
6	`Request Service (RQS) Summery Bit:` A 1 in this bit position indicates that the analyzer has at least one reason to report a status change. This bit is also called the master summary status bit (MSS).
7	`Operation Status Summary Bit:` A 1 in this bit position indicates that the operation status summary bit has been set. The operation status event register can then be read to determine the specific event that caused this bit to be set.

To query the status byte register, send the `*STB` command. The response will be the *decimal* sum of the bits that are set to 1. For example, if bit number 7 and bit number 3 are set to 1, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned.

In addition to the status byte register, the status byte group also contains the service request enable register. The status byte service request enable register lets you choose which bits in the Status Byte Register will trigger a service request.

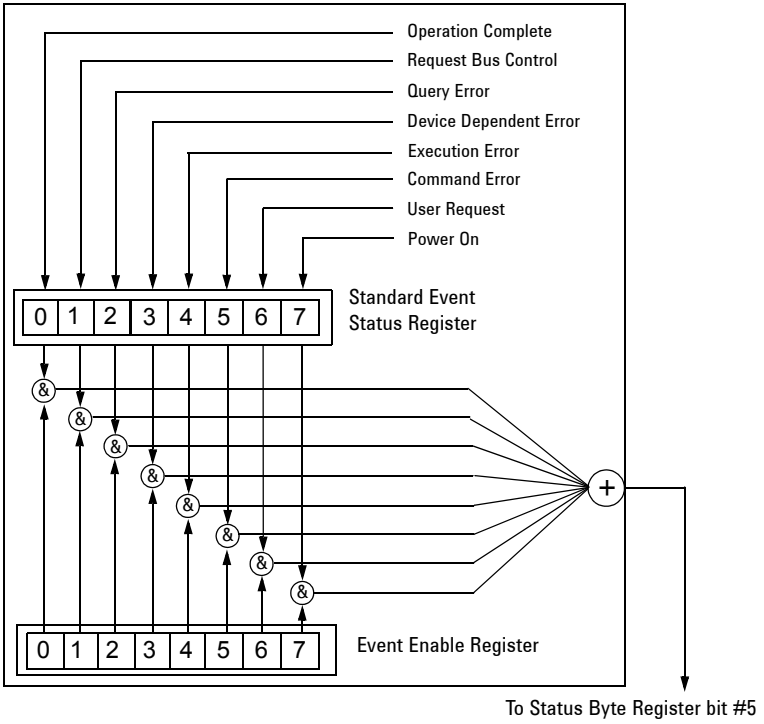
Standard Event Status Register

The standard event status register is used to determine the specific event that sets bit 5 in the status byte register. The standard event status register does *not* have negative and positive transition registers, nor a condition register. Use the IEEE common commands at the beginning of "Command Reference" on page 49 to access the register.

To query the standard event status register, send the **`*ESR`** command. The response will be the *decimal* sum of the bits which are set to 1. For example, if bit number 7 and bit number 3 are set to 1, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned.

See "Setting and Querying the Status Register" on page 25 for more information.

Figure 2 Standard Event Status Register Diagram



The standard event status register contains following bits:

Bit	Description
0	`Unused`
1	`Request Bus Control:` This bit is always set to 0. (The analyzer does not request control.)
2	`Query Error:` A 1 in this bit position indicates that a query error has occurred. Query errors have SCPI error numbers from 499 to 400.
3	`Device Dependent Error:` A 1 in this bit position indicates that a device dependent error has occurred. Device dependent errors have SCPI error numbers from –399 to –300 and 1 to 32767.
4	`Execution Error:` A 1 in this bit position indicates that an execution error has occurred. Execution errors have SCPI error numbers from –299 to –200.
5	`Command Error:` A 1 in this bit position indicates that a command error has occurred. Command errors have SCPI error numbers from –199 to –100.
6	`User Request Key (Local):` A 1 in this bit position indicates that the **[Preset/System]** (Local) key has been pressed. This is true even if the analyzer is in local lockout mode.
7	`Power On:` A 1 in this bit position indicates that the analyzer has been turned off and then on.

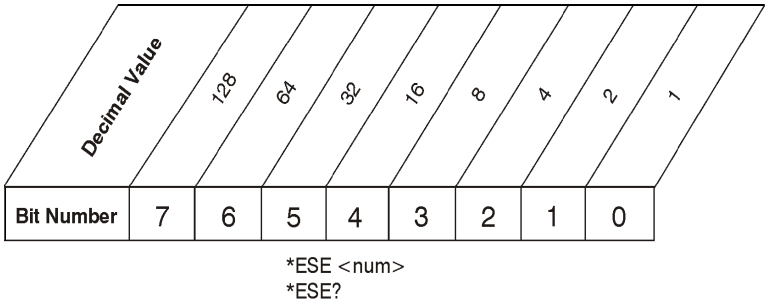
The standard event status register is used to determine the specific event that set bit 5 in the status byte register. To query the standard event status register, send the command `*ESR?`. The response will be the decimal sum of the bits which are enabled (set to 1). For example, if bit number 7 and bit number 3 are enabled, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned.

In addition to the standard event status register, the standard event status group also contains a standard event status enable register. This register lets you choose which bits in the standard event status register will set the summary bit (bit 5 of the status byte register) to 1. Send the `*ESE <integer>` command where `<integer>` is the sum of the decimal values of the bits you want to enable. For example, to enable bit 7 and bit 6 so that whenever either of those bits is set to 1, the standard event status summary bit of the status byte register will be set to 1, send the

command `*ESE 192 (128 + 64)`. The command `*ESE?` returns the decimal value of the sum of the bits previously enabled with the `*ESE <integer>` command.

The standard event status enable register presets to zeros (0).

Figure 3 Standard Event Status Event Enable Register



4b
Programming Example

This chapter provides some programming conventions and examples for your further reference.

Overview

The programming examples in this section keep to the following 3 conventions:

- The programming examples were written for use on an compatible PC.
- The programming examples use USB interface.
- The programming examples are written in C programming language and SCPI programming commands, using Agilent VISA transition library (Agilent VTL).

The Agilent VTL is installed when you installed the Agilent IO libraries suite.

The Agilent IO libraries suite contains the latest Agilent VTL and is available at:

http://www.agilent.com/find/iolib

NOTE

Agilent Technologies provides programming examples for illustration only. All sample programs assume that you are familiar with the programming language being demonstrated and the tools used to create and debug procedures.

You have a royalty-free right to use, modify, reproduce and distribute the sample application files in any way you find useful, provided that you agree that Agilent has no warranty, obligations, or liability for any sample application files.

Programming in C using the VTL

This section includes some basic information about programming in the C language using Agilent VISA transition library (VTL). Note that some of this information may not be relevant to your particular application. For example, if you are not using VXI instruments, the VXI references will not be relevant.

Typical Example Program Contents

The following table summaries the VTL function calls used in the example programs.

`visa.h`	This file is included at the beginning of the each file to provide the function prototypes and constants defined by VTL. For C and C++ programs, you must include the `visa.h` header file at the beginning of every file that contains VISA function calls: `#include "visa.h"`
`ViSession`	The ViSession is a VTL data type. Each object that will establish a communication channel must be defined as `ViSession`. Sessions must firstly be opened on the default resource manager, and then for each resource you will be using.
`viOpenDefaultRM`	You must first open a session with the default resource manager with the `viOpenDefaultRM` function, and then for each resource you will be using. This function will initialize the default resource manager and return a pointer to that resource manager session. `viOpenDefaultRM(&sesn)`

`viOpen`	This function establishes a communication channel with the device specified. A session identifier that can be used with other VTL functions is returned. This call must be made for each device you will be using.
	`viOpenDefaultRM(&sesn)`
	`viOpen(sesn, rsrcName, accessMode, timeout, &vi)`

`viPrintf` `viScanf`	These are the VTL formatted I/O functions that are patterned after those used in the C programming language. The `viPrintf` call sends the SCPI commands to the analyzer. The `viPrintf` call can also be used to query the analyzer. The `viScanf` call is then used to read the results.
`viWrite`	This function synchronously sends the data pointed to by `buf` to the device specified by `vi`. Only one synchronous write operation van occur at any one time. `viWrite(vi, buf, count, &retCount)`

`viRead`	This function synchronously reads raw data from the session specified by the `vi` parameter and stores the result in location where `buf` is pointing. Only one synchronous read operation can occur at any one time. `viRead(vi, buf, count, &retCount)`

`viClose`	This function must be used to close each session. When you close a device session, all data structures that had been allocated for the session will be set free. If you close the default resource manager session, all sessions opened using that resource manager session will be closed.
	`viClose(vi);`
	`viClose(defaultRM)`

Example Program

This example program queries a USB device for an identification string and prints the results. Note that you must change the address if something other than the default USB address value is required.

```
/*idn.c - program filename */
#include "visa.h"
#include <stdio.h>
void main ()
{
char buf[300]
ViSession viN9322C
ViStatus viSatus;
ViSession defaultRM;
/*Open session to USB device */
viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"USB0::2391::8472::000
0000000::0::INSTR",VI_NULL,VI_NULL,&viN9322C);
/*Initialize device */
viPrintf(viN9322C, "*RST\n");
/*Send an *IDN? string to the device */
printf(viN9322C, "*IDN?\n");
/*Read results */
```

```
viScanf(viN9322C, "%t", &buf);
/*Print results */
printf("Instrument identification string: %s\n",
buf);
/* Close the sessions */
viClose(viN9322C);
viClose(defaultRM);
}
```

Including the VISA Declarations File

For C and C++ programs, you must include the **`visa.h`** header file at the beginning of every file that contains VTL function calls:

```
#include "visa.h"
```

This header file contains the VISA function prototypes and the definitions for all VISA constants and error codes. The **`visa.h`** header file includes the **`visatype.h`** header file.

The `visatype.h` header file defines most of the VISA types. The VISA types are used throughout VTL to specify data types used in the functions. For example, the `viOpenDefaultRM` function requires a pointer to a parameter of type `ViSession`. If you find `ViSession` in the `visatype.h` header file, you will find that `ViSession` is eventually typed as an unsigned long.

Opening a Session

A session is a channel of communication. Sessions must first be opened on the default resource manager, and then for each device you will be using. The following is a summary of sessions that can be opened:

- A **resource manager session** is used to initialize the VISA system. It is a parent session that knows about all the opened sessions. A resource manager session must be opened before any other session can be opened.

- A **device session** is used to communicate with a device on an interface. A device session must be opened for each device you will be using. When you use a device session you can communicate without worrying about the type of interface to which it is connected. This insulation makes applications more robust and portable across interfaces. Typically a device is an instrument, but could be a computer, a plotter, or a printer.

NOTE

All devices that you will be using need to be connected and in working condition prior to the first VTL function call (`viOpenDefaultRM`). The system is configured only on the *first* `viOpenDefaultRM` per process. Therefore, if `viOpenDefaultRM` is called without devices connected and then called again when devices are connected, the devices will not be recognized. You must close **ALL** resource manager sessions and re-open with all devices connected and in working condition.

Device Sessions

There are two parts to opening a communications session with a specific device. First you must open a session to the default resource manager with the `viOpenDefaultRM` function. The first call to this function initializes the default resource manager and returns a session to that resource manager session. You only need to open the default manager session once. However, subsequent calls to `viOpenDefaultRM` returns a session to a unique session to the same default resource manager resource.

Next, you open a session with a specific device with the `viOpen` function. This function uses the session returned from `viOpenDefaultRM` and returns its own session to identify the device session. The following shows the function syntax:

```
viOpenDefaultRM (sesn);

viOpen (sesn, rsrcName, accessMode, timeout, vi);
```

The session returned from `viOpenDefaultRM` must be used in the *sesn* parameter of the `viOpen` function. The `viOpen` function then uses that session and the device address specified in the *(resource name)* parameter to open a device session. The *vi* parameter in `viOpen` returns a session identifier that can be used with other VTL functions.

Your program may have several sessions open at the same time by creating multiple session identifiers by calling the `viOpen` function multiple times.

The following summarizes the parameters in the previous function calls:

sesn	This is a session returned from the `viOpenDefaultRM` function that identifies the resource manager session.
rsrcName	This is a unique symbolic name of the device (device address).
accessMode	This parameter is not used for VTL. Use VI_NULL.
timeout	This parameter is not used for VTL. Use VI_NULL.
vi	This is a pointer to the session identifier for this particular device session. This pointer will be used to identify this device session when using other VTL functions.

Addressing a Session

As seen in the previous section, the *rsrcName* parameter in the `viOpen` function is used to identify a specific device. This parameter is made up of the VTL interface name and the device address. The interface name is determined when you run the VTL Configuration Utility. This name is usually the interface type followed by a number. The following table illustrates the format of the *rsrcName* for the different interface types:

The following describes the parameters used above:

board	This optional parameter is used if you have more than one interface of the same type. The default value for *board* is 0.
VXI logical address	This is the logical address of the VXI instrument.
primary address	This is the primary address of the USB device.
secondary address	This optional parameter is the secondary address of the USB device. If no secondary address is specified, none is assumed.
INSTR	This is an optional parameter that indicates that you are communicating with a resource that is of type **INSTR**, meaning instrument.

Closing a Session

The `viClose` function must be used to close each session. You can close the specific device session, which will free all data structures that had been allocated for the session. If you close the default resource manager session, all sessions opened using that resource manager will be closed.

Since system resources are also used when searching for resources (`viFindRsrc`) or waiting for events (`viWaitOnEvent`), the `viClose` function needs to be called to free up find lists and event contexts.

Checking the USB Connection

Usually, using "*IDN?" verifies the data transferring between the controller PC and the instrument.

```
**************************************************
#include "visa.h"
#include <stdio.h>

#define BufferSize 128

static ViStatus status;
static ViSession defaultRM;
static ViSession inst_N9322C;
static ViUInt32 rcount;
static unsigned char buffer[BufferSize];

int main(void)
{
    /* Connect N9322C and read its "IDN". */
    status = viOpenDefaultRM (&defaultRM);
    status = viOpen (defaultRM,
"USB0::2391::8472::0000000000::0::INSTR", VI_NULL,
VI_NULL, &inst_N9322C);
    if (status != VI_SUCCESS)
    return -1; //failed to connect N9322C/
   /* Read "IDN" from N9322C" */
    status = viWrite (inst_N9322C, "*RST\n",
StringLength("*RST\n"), &rcount);
    status = viWrite (inst_N9322C, "*IDN?\n",
StringLength("*IDN?\n"), &rcount);
    status = viRead (inst_N9322C, buffer, BufferSize,
&rcount);
    /* Close connection to N9322C. */
    status = viClose (inst_N9322C);
    status = viClose (defaultRM); return 1;
}
```

Using C with Marker Peak Search and Peak Excursion

```
/*************************************************************/
/* Using Marker Peak Search and Peak Excursion */
/* */
/* This example is for the N9322C Handheld Spectrum Analyzer.  */
/* */
/* This C programming example does the following. */
/* The SCPI instrument commands used are given as reference. */
/* */
/* - Opens a USB session */
/* - Clears the Analyzer */
/* *CLS */
/* - Resets the Analyzer */
/* *RST */
/* - Sets the analyzer center frequency, span and units */
/* SENS:FREQ:CENT freq */
/* SENS:FREQ:SPAN freq */
/* UNIT:POW DBM */
/* - Set the input port to the 50 MHz amplitude reference */
/* CAL:SOUR:STAT ON */
/* - Set the analyzer to single sweep mode */
/* INIT:CONT 0 */
/* - Prompt the user for peak excursion and set them */
/* CALC:MARK:PEAK:EXC dB */
/* - Set the peak threshold to -90 dBm */
/* CALC:MARK:PEAK:THR:STAT ON */
/* CALC:MARK:PEAK:THR <ampl> */
/* - Trigger a sweep and delay for sweep to complete */
/* INIT:IMM */
/* - Set the marker to the maximum peak */
/* CALC:MARK1:MAX */
/* - Query and read the marker frequency and amplitude */
/* CALC:MARK:X? */
/* CALC:MARK:Y? */
/* - Close the session */
/*************************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

ViSession defaultRM, viN9322C;
ViStatus errStatus;
ViChar cIdBuff[256]= {0};
char cEnter = 0;
int iResult = 0;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{

viQueryf(viN9322C, "*IDN?\n", "%t", &cIdBuff);
/* prompt the user*/
/* to connect the amplitude reference output to the input*/
printf ("Connect CAL OUT to the RF IN \n");
printf (".....Press Return to continue \n");
scanf( "%c",&cEnter);
/*Externally route the 50MHz Signal*/
viPrintf(viN9322C,"CAL:SOUR:STAT ON \n");

}
void main()
{
/*Program Variables*/
ViStatus viStatus = 0;
double dMarkerFreq = 0;
double dMarkerAmpl = 0;
float fPeakExcursion =0;
```

```
/*Open a USB session.*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"USB0::2391::8472::0000000000::0::INSTR",V
I_NULL,VI_NULL,&viN9322C);
if(viStatus)
{
printf("Could not open a session to USB device\n");
exit(0);
}
/*Clear the instrument*/
viClear(viN9322C);

/*Reset the instrument*/
viPrintf(viN9322C,"*RST\n");

/*Set Y-Axis units to dBm*/
viPrintf(viN9322C,"UNIT:POW DBM\n");

/*Set the analyzer center frequency to 50MHZ*/
viPrintf(viN9322C,"SENS:FREQ:CENT 50e6\n");

/*Set the analyzer span to 50MHZ*/
viPrintf(viN9322C,"SENS:FREQ:SPAN 50e6\n");

/*Display the program heading */
printf("\n\t\t Marker Program \n\n" );

/* Check for the instrument model number and route the 50MHz signal
accordingly*/
Route50MHzSignal();

/*Set analyzer to single sweep mode*/
viPrintf(viN9322C,"INIT:CONT 0 \n ");

/*User enters the peak excursion value*/
printf("\t Enter PEAK EXCURSION in dB: ");
scanf( "%f",&fPeakExcursion);
```

```
/*Set the peak excursion*/
viPrintf(viN9322C,"CALC:MARK:PEAK:EXC %1fDB \n",fPeakExcursion);

/*Set the peak thresold */
viPrintf(viN9322C,"CALC:MARK:PEAK:THR -90 \n");

/*Trigger a sweep and wait for completion*/
viPrintf(viN9322C,"INIT:IMM \n");

/*Set the marker to the maximum peak*/
viPrintf(viN9322C,"CALC:MARK:MAX \n");

/*Query and read the marker frequency*/
viQueryf(viN9322C,"CALC:MARK:X? \n","%lf",&dMarkerFreq);
printf("\n\t RESULT: Marker Frequency is: %lf MHZ \n\
n",dMarkerFreq/10e5);

/*Query and read the marker amplitude*/
viQueryf(viN9322C,"CALC:MARK:Y?\n","%lf",&dMarkerAmpl);
printf("\t RESULT: Marker Amplitude is: %lf dBm \n\n",dMarkerAmpl);

/*Close the session*/
viClose(viN9322C);
viClose(defaultRM);
}
```

Using Marker Delta Mode and Marker Minimum Search

```
/*************************************************************/
/* Using Marker Delta Mode and Marker Minimum Search */
/* */
/* This example is for the N9322C Spectrum Analyzers */
/* */
/* This C programming example does the following. */
/* The SCPI instrument commands used are given as reference. */
/* */
/* - Opens a USB session */
/* - Clears the Analyzer */
/* - Resets the Analyzer */
/* *RST */
/* - Set the input port to the 50 MHz amplitude reference */
/* CAL:SOUR:STAT ON */
/* - Set the analyzer to single sweep mode */
/* INIT:CONT 0 */
/* - Prompts the user for the start and stop frequencies */
/* - Sets the start and stop frequencies */
/* SENS:FREQ:START freq */
/* SENS:FREQ:STOP freq */
/* - Trigger a sweep and delay for sweep completion */
/* INIT:IMM */
/* - Set the marker to the maximum peak */
/* CALC:MARK:MAX */
/* - Set the analyzer to activate the delta marker */
/* CALC:MARK:MODE DELT */
/* - Trigger a sweep and delay for sweep completion */
/* INIT:IMM */
/* - Set the marker to the minimum amplitude search mode */
/* CALC:MARK:PEAK:SEAR:MODE MIN */
/* - Set the marker to the minimum peak */
/* CALC:MARK:MAX */
/* - Query and read the marker amplitude */
/* CALC:MARK:Y? */
/* - Close the session */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

ViSession defaultRM, viN9322C;
ViStatus errStatus;
ViChar cIdBuff[256] ={0};
char cEnter = 0;
int iResult =0;

/*Set the input port to the 50MHz amplitude reference*/
void Route50MHzSignal()
{
       viQueryf(viN9322C, "*IDN?\n", "%t", &cIdBuff);
       /* prompt the user*/
       /* to connect the amplitude reference output to the
input*/
       printf ("Connect CAL OUT to the RF IN \n");
       printf (".....Press Return to continue \n");
       scanf( "%c",&cEnter);
       /*Externally route the 50MHz Signal*/
       viPrintf(viN9322C,"CAL:SOUR:STAT ON \n");
}

void main()
{
       /*Program Variable*/
       ViStatus viStatus = 0;
       double dStartFreq =0.0;
       double dStopFreq =0.0;
       double dMarkerAmplitude = 0.0;
```

```
        {
         /* Open an USB session*/
        viStatus=viOpenDefaultRM(&defaultRM);
        viStatus=viO-
pen(defaultRM,"USB0::2391::8472::9876543210::0::INSTR",VI_NULL,V
I_NULL,&viN9322C);
        if(viStatus)

               printf("Could not open a session to USB device!\n");
               exit(0);
        }
        /*Clear the instrument*/
        viClear(viN9322C);

        /*Reset the instrument*/
        viPrintf(viN9322C,"*RST\n");
        /*Display the program heading */
        printf("\n\t\t Marker Delta Program \n\n" );

        /*Check for the instrument model number and route the 50MHz
signal accordingly*/
        Route50MHzSignal();

        /*Set the analyzer to single sweep mode*/
        viPrintf(viN9322C,"INIT:CONT 0\n");

        /*Prompt the user for the start frequency*/
        printf("\t Enter the Start frequency in MHz ");

        /*The user enters the start frequency*/
        scanf("%lf",&dStartFreq);

        /*Prompt the user for the stop frequency*/
        printf("\t Enter the Stop frequency in MHz ");

        /*The user enters the stop frequency*/
        scanf("%lf",&dStopFreq);
```

```
/*Set the analyzer to the values given by the user*/
       //viPrintf(viN9322C,"SENS:FREQ:STAR %lf
//MHZ;:SENS:FREQ:STOP %lf MHZ\n",dStartFreq,dStopFreq);
       viPrintf(viN9322C,":SENS:FREQ:STAR %lf MHz\n",dStart-
Freq);
       viPrintf(viN9322C,":SENS:FREQ:STOP %lf MHZ\n",dStopFreq);

       /*Trigger a sweep, delay for completion*/
       viPrintf(viN9322C,"INIT:IMM\n");
       //delay(1);

       /*Set the marker to the maximum peak*/
       viPrintf(viN9322C,"CALC:MARK:MAX\n");

       /*Set the analyzer to activate delta marker mode*/
       viPrintf(viN9322C,"CALC:MARK:MODE DELT\n");

       /*Trigger a sweep, delay for completion*
       viPrintf(viN9322C,"INIT:IMM\n");
       Sleep(1);

       /*Set the marker to minimum amplitude Search mode*/
       viPrintf(viN9322C,"CALC:MARK:PEAK:SEAR:MODE MIN\n");

       /*Set the marker to minimum amplitude*/
       viPrintf(viN9322C,"CALC:MARK:MAX\n");

       /*Query and read the marker amplitude*/
       viQueryf(viN9322C,"CALC:MARK:Y?\n","%lf",&dMarkerAmpli-
tude);

       /*print the marker amplitude*/
       printf("\n\n\tRESULT: Marker Amplitude Delta =%lf dB\n\
n",dMarkerAmplitude);

       /*Close the session*/
       viClose(viN9322C);
```

4 Programming Example

5 Command Reference

This chapter contains SCPI (Standard Commands for Programmable Instruments) programming commands for the spectrum analyzer core operation.

IEEE Common Commands

The first few pages of this chapter contain common commands specified in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std. 488.1-1987*. New York, NY, 1992.

Following these commands, the Agilent N9322C spectrum analyzers SCPI commands are listed.

Clear Status

`*CLS`

Clears the status byte register. It does this by emptying the error queue and clearing all bits in all of the event registers. The status byte register summarizes the states of the other registers. It is also responsible for generating service requests.

Remark: See `*STB?`

Standard Event Status Enable

`*ESE <number>`
`*ESE?`

Sets the bits in the standard event status enable register. This register monitors I/O errors and synchronization conditions such as operation complete, request control, query error, device dependent error, execution error, command error and power on. A summary bit is generated on execution of the command.

The query returns the state of the standard event status enable register.

Range: Integer, 0 to 255

Example: `*ESE 36` Enables the Standard Event Status Register to monitor query and command errors (bits 2 and 5).

`*ESE?` Returns a 36 indicating that the query and command status bits are enabled.

Standard Event Status Register Query

```
*ESR?
```

Queries and clears the standard event status event register. (This is a destructive read.) The value returned reflects the current state (0/1) of all the bits in the register.

Range: Integer, 0 to 255

Example: `*ESR?` returns a 1 if there is either a query or command error, otherwise it returns a zero.

Identification Query

```
*IDN?
```

Returns an instrument identification information string. The string will contain the model number, serial number and firmware revision. The response is organized into four fields separated by commas. The field definitions are manufacturer, model, serial number and software version.

Example: `*IDN?` returns instrument information, such as:
`Agilent Technologies, N9322C, 45310116, A.04.20`

Key access: **SYS > More > Show System**

Operation Complete Query

```
*OPC
*OPC?
```

Sets bit 0 in the standard event status register to “1” when all pending operations have finished.

The query stops any new commands from being processed until the current processing is complete. Then it returns a “1”, and the program continues. This query can be used to synchronize events of other instruments on the external bus.

Returns a “1” if the last processing is complete. Use this query when there’s a need to monitor the command execution status, such as a sweep execution.

Reset

```
*RST
```

This command presets the instrument to a factory defined condition that is appropriate for remote programming operation. `*RST` is equivalent to performing the two commands :`SYSTem:PRESet` and `*CLS`. This command always performs a factory preset.

NOTE The preset performed by `*RST` is always a factory preset. That is, the same preset performed by `:SYSTem:PRESet` when `:SYSTem:PRESet:TYPE` is set to `DFT`

Key access: **Preset**

Service Request Enable

```
*SRE <integer>
*SRE?
```

This command enables the desired bits of the service request enable register.

The query returns the value of the register, indicating which bits are currently enabled. The default value is 255.

Example: `*SRE 16` enables bits 4 in the service request enable register.

Range: Integer, 0 to 255

Status Byte Query

```
*STB?
```

Returns the value of the status byte register without erasing its contents.

Range: Integer, 0 to 255

Example: If a 16 is returned, it indicates that bit 5 is set and one of the conditions monitored in the standard event status register is set.

Self Test Query

`*TST?`

This query is used by some instruments for a self test.

Range: Integer, 0 to 255

Wait-to-Continue

`*WAI`

This command causes the instrument to wait until all pending commands are completed before executing any additional commands. There is no query form to the command.

Range: Integer, 0 to 255

System Subsystem

This subsystem is used to set the controls and parameters associated with the overall system communication. These functions are not related to instrument performance.

Screen Brightness Setting

```
:SYSTem:SCReen:BRIGhtness <value>
:SYSTem:SCReen:BRIGhtness?
```

This command sets the screen brightness of the instrument.

Range: 0 to 10

Example: `:SYSTem:SCReen:BRIGhtness 4`

Key access: **System> Screen Setting**

Auto Screen Brightness State

```
:SYSTem:SCReen:BRIGhtness:AUTO OFF|ON|0|1
:SYSTem:SCReen:BRIGhtness:AUTO?
```

This command turns on/off auto screen brightness state.

Example: `:SYSTem:SCReen:BRIGhtness:AUTO ON`

Key access: **System> Screen Setting**

Set System Time

```
:SYSTem:TIME <"hhmmss">
:SYSTem:TIME?
```

Sets the system time of the instrument.

Hour must be an integer 0 to 23.

Minute must be an integer 0 to 59.

Second must be an integer 0 to 59.

Example: `:SYSTem:TIME "150233"`

Key access: **System> Time/Date > Time**

Set System Date

```
:SYSTem:DATE <"yyyymmdd">
:SYSTem:DATE?
```

Sets the system date of the real-time clock of the instrument. Year is a 4-digit integer. Month is an integer 1 to 12. Day is an integer 1 to 31 (depending on the month)

Example: `:SYSTem:DATE "20120912"`

Key access: **System > Time/Date > Date**

System Time Zone Offset

```
:SYSTem:TIME:ZONE <value>
:SYSTem:TIME:ZONE?
```

Sets the time zone offset. The offset is set according to the Greenwich Mean Time. Below example sets the time zone offset to 8 hours.

Example: `:SYSTem:TIME:ZONE 8`

Key access: **System > Time/Date > Time Zone Offset**

Run Lasted Time Query

```
:SYSTem:PON:TIME?
```

This query returns the time that has elapsed since the analyzer was last turned on.

Language Type

```
:SYSTem:LANGuage
SCHINESE|TCHINESE|ENGLISH|FRENCH|GERMAN|ITALIAN|
JAPANESE|KOREAN|PORTUGUESE|RUSSIAN|SPANISH

:SYSTem:LANGuage?
```

Sets the language type for the soft key menu display. The language type setting is included in the setup(*.SET) file.

Example: `:SYSTem:LANGuage SCHINESE`

Key access: **System > Language Type**

Enable Option

```
:SYSTem:LKEY <"option">,<"license key">
```

Use this command to enable the specified option with the license key.

Example: `:SYSTem:LKEY "PA3","ABCDEFGH"`

Key access: **System > More > Service > Add Option**

Disable Option

```
:SYSTem:LKEY:DISable <"license key">
```

Use this command to disable the installed options.

Example: `:SYSTem:LKEY: DISable "ABCDEFGH"`

Key access: **System > System Info > Installed Options > Delete License**

Installed Options Query

```
:SYSTem:OPTions?
```

This command returns a list of the options that are installed.

Key access: **System > System Info > Installed Options**

System Error

```
:SYSTem:ERRor[:NEXT]?
```

Use this command to read the system error information.

Example: `:SYSTem:ERRor?`

Key access: **System > System Info > Error History**

Calibrate Time Base by External Reference Signal

```
:SYSTem:CALibration:TBASe:EXT
```

Use this command to calibrate the time base by external signal. Please connect a BNC cable with 10 MHz reference signal to the **EXT TRIG IN** connector before using this command.

Key access: **System > More > Service > Calibration > Time Base By EXT**

Query Time Base Calibration Result

```
:SYSTem:CALibration:TBASe:FREQuency?
```

Use this command to query the time base calibration result.

Calibration Amplitude

```
:CALibration
```

Use this command to trigger the amplitude calibration. Connect the amplitude calibration reference output port to the RF input port, and turn on the amplitude calibration reference output

Example: `:CALibration`

Key access: **System > More > Service > Calibration > Amp Alignment**

Amplitude Calibration Reference Signal Output

```
:CALibration:SOURce:STATe ON|OFF|1|0:
:CALibration:SOURce:STATe?
```

Use this command to turn on/off the amplitude calibration reference output.

Example: `:CALibration`

Key access: **System > More > Service > Calibration > Cal Out On/off**

Query Amplitude Calibration Result

```
:SYSTem:CALibration:AMPLitude:STATe?
```

Use this command to query the amplitude calibration result

BNC Port Input Type Setting

```
:SYSTem:PORT:EXTInput REF|TRIGger
:SYSTem:PORT:EXTInput?
```

Toggles the external input between 10 MHz reference signal and TTL signal.

Key access: **System > More > Port Setting > Ext Input**

Probe Power

```
:SYSTem:PORT:PROBe OFF|ON|0|1
:SYSTem:PORT:PROBe?
```

This command turns on/off the probe power port on the top panel.

Key access: **System > More > Port Setting > Probe Power**

IP Config Host Name

```
:SYSTem:PORT:IP:HNAMe <"string">
:SYSTem:PORT:IP:HNAMe?
```

Sets a host name for the analyzer in network.

Key access: **System > More > Port Setting > IP Admin > Sys Name**

IP Address

```
:SYSTem:PORT:IP:ADDRess <"xxx.xxx.xxx.xxx">
:SYSTem:PORT:IP:ADDRess?
```

Sets a host name for the analyzer in network.

Example: :SYSTem:PORT:IP:ADDRess "192.168.0.113"

Key access: **System > More > Port Setting > IP Admin >IP Address**

IP Address Assignment

```
:SYSTem:PORT:IP:ADDRess:TYPE STATIC|DHCP
:SYSTem:PORT:IP:ADDRess:TYPE?
```

Toggles the IP assignment setting between static (manual) and DHCP (dynamic assignment) mode.

Key access: **System > More > Port Setting > IP Admin > IP Address**

Gateway Setting

```
:SYSTem:PORT:IP:GWAY <"xxx.xxx.xxx.xxx">
:SYSTem:PORT:IP:GWAY?
```

Sets the gateway for the analyzer in the network. The gateway will be fetched automatically if the IP assignment is set to DHCP.

Example: `:SYSTem:PORT:IP:GWAY "192.168.0.1"`

Key access: **System > More > Port Setting > IP Admin > Gateway**

Subnet Mask

```
:SYSTem:PORT:IP:SMASk <"xxx.xxx.xxx.xxx">
:SYSTem:PORT:IP:SMASk?
```

Sets the subnet mask according to the PC network settings. The subnet mask will be set automatically if the IP assignment is set to DHCP.

Example: `:SYSTem:PORT:IP:SMASk "255.255.255.1"`

Key access: **System > More > Port Setting > IP Admin > Subnet Mask**

IP Config Apply

```
:SYSTem:PORT:IP:APPLy
```

Use this command to apply all the IP settings according to the IP assignment settings. If the IP assignment is set to DHCP, the IP address, gateway, and subnet mask will be set automatically.

Key access: **System > More > Port Setting > IP Admin > Apply**

Erase Memory

```
:SYSTem:SECurity:ERASe
```

Use this command to erase all the user data saved in internal memory. This command is only available when the option SEC is installed.

Key access: **System > More > Securities > Erase Memory**

System Preset

```
:SYSTem:PRESet
```

Use this command to preset the instrument. The preset type is based on the setting of Preset Type: DFT, User or Last.

Key access: **Preset**

Factory Default

```
:SYSTem:FDEFault
```

Set both the measure and setting parameters to factory preset parameters.

Key access: **System > More > Factory Default**

Timed Power On

```
:SYSTem:TIMed:PON:STATe OFF|ON|0|1
:SYSTem:TIMed:PON:STATe?
```

Use this command turn on/off the timed power-on function. The analyzer will be turned on in a user-defined time.

Example:
```
:SYSTem:TIMed:PON:STATe ON
```

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr On > Power On On/Off**

Timed Power On Repeat Mode

```
:SYSTem:TIMed:PON:MODE ONCE|EVERYDAY
:SYSTem:TIMed:PON:MODE?
```

Timed powers on the analyzer just once or repeat everyday.

Example:
```
:SYSTem:TIMed:PON:MODE ONCE
```

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr On > Repeat Mode**

Time Power On Time

```
:SYSTem:TIMed:PON:TIME <"HHMMSS">
:SYSTem:TIMed:PON:TIME?
```

Sets the power on time, then the analyzer will be turned on automatically.

Example: `:SYSTem:TIMed:PON:TIME "122332"`

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr On > Time**

Time Power On Date

```
:SYSTem:TIMed:PON:DATE <"YYYMMDD">
:SYSTem:TIMed:PON:DATE?
```

Sets the power on date. This commmand is only available when the repeat mode is set to once.

Example: `:SYSTem:TIMed:PON:DATE "20120922"`

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr On > Date**

Timed Power Off State

```
:SYSTem:TIMed:POFF:STATE OFF|ON|0|1
:SYSTem:TIMed:POFF:STATE?
```

This command turns on/off timed power off status of instrument.

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr Off > Power Off**

Timed Power Off Repeat Mode

```
:SYSTem:TIMed:POFF:MODE ONCE|EVERYDAY
:SYSTem:TIMed:POFF:MODE?
```

Use this command to toggle timed power off mode between once and everyday in a user-defined time

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr Off > Repeat Mode Once/Everyday**

Time Power Off Time

```
:SYSTem:TIMed:POFF:TIME <"HHMMSS">
:SYSTem:TIMed:POFF:TIME?
```

Sets the power off time of the instrument.

Example: `:SYSTem:TIMed:POFF:TIME "122332"`

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr Off > Time**

Time Power Off Date

```
:SYSTem:TIMed:POFF:DATE <"YYYMMDD">
:SYSTem:TIMed:POFF:DATE?
```

Sets the power off date. It's available when repeat mode is set to once.

Example: `:SYSTem:TIMed:POFF:DATE "20120922"`

Key access: **Shift > System > Pwr On/Off Preset > Time Pwr Off > Date**

Power On Type

```
:SYSTem:PON:TYPE DFT|USER|LAST
:SYSTem:PON:TYPE?
```

Uses this command to preset the analyzer to default, user, or last state.

Example: `:SYSTem:PON:TYPE USER`

Key access: **Shift > System > Pwr On/Off Preset > Pwr On Setting**

Preset Type

```
:SYSTem:PRESet:TYPE DFT|USER|LAST
:SYSTem:PRESet:TYPE?
```

Uses command to set analyzer to power on in default, user, or last state.

Example: `:SYSTem:PRESet:TYPE USER`

Key access: **System > Pwr On/Off Preset > Preset Type**

Save User Preset

```
:SYSTem:PRESet[:USER]:SAVE
```

Uses this command to save the current instrument state as the user state for the power on setting and preset type.

Key access: **System > Pwr On/Off Preset > Save User**

Low Frequency Channel Status

```
:SYSTem:CONFigure:LFCHannel OFF|ON|0|1
:SYSTem:CONFigure:LFCHannel?
```

Use this command to turn on/off the low frequency channel status.

Key access: **Freq > More > Low Frequency**

Power Reset

```
:SYSTem:POWer:RESet
```

Use this command to restart the instrument.

Power Off

```
:SYSTem:POWer:OFF
```

Use this command to turn off the instrument.

Speaker Volume

```
:SYSTem:SPEaker <value>
:SYSTem:SPEaker?
```

Use this command to adjust the volume of the speaker.

Example: `:SYSTem:SPEaker 20`

Speaker Volume Flag

```
:SYSTem:SPEaker:STATe OFF|ON|0|1
```

```
:SYSTem:SPEaker:STATe?
```

Use this command to turn on the speaker in the instrument.

Example:
```
:SYSTem:SPEaker:STATe?
```

Hardware Message

```
:SYSTem:CONFigure:HARDware?
```

Use this command to query the hardware message of the instrument.

Software Message

```
:SYSTem:CONFigure:SOFTware?
```

Use this command to query the software message of the instrument.

System Message

```
:SYSTem:CONFigure:SYSTem?
```

Use this command to query the system message of the instrument.

Example:
```
:SYSTem:CONFigure:SYSTem?
```

Memory Subsystem

The Memory subsystem provides access to mass storage devices such as internal or external disk drives. It is corresponding to the front panel file submenu.

NOTE The catalog C:\ indicates the root directory of the internal memory of analyzer. The catalog E:\ indicates the root directory of the USB memory stick.

Catalog the Selected Memory Location

```
:MMEMory:CATalog? <"dir_path">
```

Lists all files in the specified path. The return data will be of the format:

```
<mem_used(kByte)>, <mem_free(kByte)>, <total Items>, <item_listing>.
```

Each < item listing> indicates the name, type, and size of each item list:

```
<name>, <type>, <size(Byte)>, <modified time>.
```

"C:\" is N9322C internal memory address, and "E:\" is the external USB storage device address.

Example: `:MMEMory:CATalog? "C:\"`

Key access: **Shift >File**

Copy a File

```
:MMEMory:COPY <file_name1>,<file_name2>
```

This command is used to copy a file. The source file name is <file_name1> and the destination file name is <file_name2>. "C:\" is N9322C internal memory address, and "E:\" is the external USB storage device address.

Example: `:MMEMory:COPY "C:\N932X.jpg", "E:\N9322C.jpg"`

Key access: **Shift > File > Files Operation > Copy To**

Move Data to File

```
:MMEMory:DATA <file_name>,<definite_length_block>
:MMEMory:DATA? <file_name>
```

Loads <definite_length_block> into the memory location <file_name>. The query returns the contents of the <file_name> in the format of a definite length block. This command can be used for copying files out of the analyzer over the remote bus.

Example: `If want to load string "abcd" into file C:\source.txt, use below command.`

:MMEM:DATA "C:\source.txt",#14abcd

Delete a File

```
:MMEMory:DELete <"file_name">
```

To delete a file. If <file_name> does not exist, a *File Name Error* will occur.

Example: `:MMEMory:DELete "C:\ABC.TRC"`

Key access: **Shift > File > Files Operation > Delete**

Load a File

```
:MMEMory:LOAD
TRC|STA|SET|LIM|ANT|STD|COR,<"file_name">
```

To load a file. If <file_name> does not exist, a *File Name Error* will occur.

Example: `:MMEMory:LOAD TRC,"C:\N932X_1.TRC"`

Key access: **Shift > File > Recall**

Store a File

```
:MMEMory:STORe
TRC|STA|SET|LIM|ANT|STD|COR,<"file_name">
```

To load a file. DAT|STA|SET||JPG|BMP|CSV indicates the file type.

Example: `:MMEMory:STORe STA,"C:\ABC.STA"`

Key access: **Shift > File > Save As**

Store Peak Table to CSV File

```
:MMEMory:STORe:PEAK <CSV file_name>
```

To store the current peak table as a CSV file. The suffix of the file name must be *.csv.

Example: `:MMEMory:STORe:PEAK "C:\ABC.CSV"`

Key access: **Shift > Peak > More > Peak Table > Export Table to CSV**

Create A New Directory

```
:MMEMory:MDIRectory <"dir_path">
```

Use this command to create a new directory to the N9322C internal memory or external USB memory.

Example: `:MMEMory:MDIRectory "C:\User"`

Key access: **Shift > File > Directory > Create Folder**

Delete A Directory

```
:MMEMory:RDIRectory <"dir_path">
```

Use this command to delete a directory from the internal memory.

Example: `:MMEMory:RDIRectory "C:\USER"`

Key access: **Shift > File > Directory > Delete Folder**

Rename A Directory or File

```
:MMEMory:REName <oldname>,<newname>
```

To rename a directory or a file.

Example: `:MMEMory:REName "C:\USER","C:\USER1"`

Key access: **Shift > File > Directory > Rename Folder**

Load Correction File

```
:MMEMory:LOAD:CORRection
ANTenna|CABLe|OTHer|USER, "oldname.cor"
```

This command loads correction file saved on the instrument.

Example: `:MMEMory:LOAD:CORRection ANTenn,"oldname.cor"`

Key access: **Amptd > More > Correction > Correction 1 > Load User, then choose a file with knob and “Enter” button**

Instrument Subsystem

The instrument subsystem includes commands selecting the instrument modes or power measurement mode.

Instrument Mode Switch

```
:INSTrument[:SELect] SA|TGENerator|MA|POWmeter|CAT
:INSTrument[:SELect]?
```

This command selects the instrument mode of N9322C. All the instrument modes are under Mode list of N9322C unit.

SA - Spectrum Analyzer mode.

TGENerator - Tracking Generator mode.

MA - Modulation Analysis mode.

POWmeter - Power Meter mode.

CAT - Reflection Measurement mode

***RST:** SA

Example: :INSTrument TGENerator

Key access: **MODE**

Measurement Mode Switch

```
:INSTrument:MEASure OFF|CHPower|ACPR|OBW|SPECtrogram|SEM|CHScanner
:INSTrument:MEASure?
```

This command selects the specific power measurement mode.

CHPower - Channel Power Measurement

ACPR - Adjacent Channel Power Ratio Measurement

OBW - Occupied Bandwidth Measurement

SPECtrogram - Spectrum Monitor

SEM - Spectrum Emission Mask

CHScanner - Channel Scanner

Example: :INSTrument:MEASure CHPower

Key access: **MEAS**

Sense Subsystem

The Sense Subsystem provides you the SCPI command reference for normal spectrum analyzer function. It is used to set the spectrum analyzer parameters such as frequency, span, attenuation and detector.

Frequency Subsection

Center Frequency

```
[:SENSe]:FREQuency:CENTer <freq>
[:SENSe]:FREQuency:CENTer UP|DOWN
[:SENSe]:FREQuency:CENTer?
```

Set the center frequency of the spectrum analyzer.

***RST:** 3.5 GHz

Example: `:FREQuency:CENTer 2 GHZ`

Key access: **Freq > Center Freq**

Center Frequency Channel

```
[:SENSe]:FREQuency:CENTer:CHANnel <channel>
[:SENSe]:FREQuency:CENTer:CHANnel?
```

Set the center frequency channel of the spectrum analyzer.

Example: `:FREQuency:CENTer:CHANnel 100`

Key access: **Freq > Unit >Channel, then Center Freq**

Start Frequency

```
[:SENSe]:FREQuency:STARt <freq>
[:SENSe]:FREQuency:STARt?
```

Set the start frequency of the spectrum analyzer.

***RST:** 0 Hz

Example: `:FREQuency:STARt 3 GHZ`

Key access: **Freq > Start Freq**

Stop Frequency

```
[:SENSe]:FREQuency:STOP <freq>
[:SENSe]:FREQuency:STOP?
```

Set the stop frequency of the spectrum analyzer.

***RST:** 7.0 GHz

Default Unit: Hz

Example: `:FREQuency:STOP 3 GHZ`

Key access: **Freq > Stop Freq**

Center Frequency Step

```
[:SENSe]:FREQuency:CENTer:STEP[:INCRement] <freq>
[:SENSe]:FREQuency:CENTer:STEP[:INCRement]?
```

Specifies the center frequency step size.

***RST:** 300 MHz

Example: `:FREQuency:CENTer:STEP 2 GHz`

Key access: **Freq > CF Step Man**

Center Frequency Channel Step

```
[:SENSe]:FREQuency:CENTer:STEP[:INCRement]:CHANnel
<channel>
[:SENSe]:FREQuency:CENTer:STEP[:INCRement]:CHANnel?
```

Specifies the center frequency Channel Step.

Example: `:FREQuency:CENTer:STEP:CHANnel 3`

Key access: **Freq > More > CF Step Man**

Frequency Unit

```
[:SENSe]:FREQuency:UNIT FREQuency|CHANnel
[:SENSe]:FREQuency:UNIT?
```

Toggle the frequency unit between frequency and chanel.

Example: `:FREQuency:UNIT CHANnel`

Key access: **Freq > Unit**

Frequency Offset

```
[:SENSe]:FREQuency:OFFSet <freq>
[:SENSe]:FREQuency:OFFSet?
```

Sets the frequency offset.

***RST:** 0.0 Hz

Example:
```
:FREQuency:OFFSet 1 GHZ
```

Key access: **Freq > More > Freq Offset**

Center Frequency Step Size Automatic

```
[:SENSe]:FREQuency:CENTer:STEP:AUTO OFF|ON|0|1
[:SENSe]:FREQuency:CENTer:STEP:AUTO?
```

Specifies whether the step size is set automatically based on the span.

***RST:** On

Key access: **Freq > CF Step Auto/Man**

Frequency Span

```
[:SENSe]:FREQuency:SPAN <freq>
[:SENSe]:FREQuency:SPAN?
```

Set the frequency span. Setting the span to 0 Hz puts the analyzer into zero span.

***RST:** 7.0 GHz

Example:
```
:FREQuency:SPAN 2 GHZ
```

Key access: **Span**

Span Channel

```
[:SENSe]:FREQuency:SPAN:CHANnel <Channel>
[:SENSe]:FREQuency:SPAN:CHANnel?
```

Set the span channel when the frequency unit is set to channel.

Example:
```
:FREQuency:SPAN:CHANnel 20
```

Key access: **Span**

Full Frequency Span

`[:SENSe]:FREQuency:SPAN:FULL`

Set the frequency span to full scale.

***RST:** 3.0 GHz

Example: `:FREQuency:SPAN:FULL`

Key access: **Span > Full Span**

Zero Span

`[:SENSe]:FREQuency:SPAN:ZERO`

Set the frequency span to zero span.

Key access: **Span > Zero Span**

Last Frequency Span

`[:SENSe]:FREQuency:SPAN:PREVious`

Set the frequency span to the previous span setting.

Key access: **Span > Last Span**

Auto Tune

`[:SENSe]:FREQuency:TUNE:IMMediate`

Auto tune the spectrum analyzer parameter to display the main signal.

Example: `:FREQuency:TUNE:IMMediate`

Key access: **Freq > Auto Tune**

Base Band Channel Switch

`[:SENSe]:FREQuency:LFCHannel OFF|ON|0|1`
`[:SENSe]:FREQuency:LFCHannel?`

Use this command to turn on/off low frequency channel.

Example: `:FREQuency:LFCHannel ON`

Key access: **Freq > More > LowFreqChannel > ON/Off**

Amplitude Subsection

Reference level

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel: <value>
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel?
```

This command sets the reference level for the Y-axis.

***RST:** 0 dB

Range: –140.00 to +20.00 dBm

Example: `:DISPlay:WINDow:TRACe:Y:RLEVel 10dBm`

Key access: **Amptd > Ref Level**

Attenuation

```
[:SENSe]:POWer[:RF]:ATTenuation <value>
[:SENSe]:POWer[:RF]:ATTenuation?
```

Set the input attenuator of the spectrum analyzer.

Range: 0 to 50 dB

Example: `:POWer:ATTenuation 30dB`

Key access: **Amptd > Attenuation**

Input Port Attenuator Auto State

```
[:SENSe]:POWer[:RF]:ATTenuation:AUTO OFF|ON|0|1
[:SENSe]:POWer[:RF]:ATTenuation:AUTO?
```

This command turns on/off auto input port attenuator state.

***RST:** On

Example: `:POWer:ATTenuation:AUTO?`

Key access: **Amptd > Attenuation > Auto**

Input Port Preamplifer

```
[:SENSe]:POWer[:RF]:GAIN[:STATe] OFF|ON|0|1
[:SENSe]:POWer[:RF]:GAIN[:STATe]?
```

Turns the internal preamp on/off.

***RST:** Off

Remarks: This command is only available when PA7 option is installed.

Example:
```
:POWer:GAIN ON
```

Key access: **AMPTD > Preamp On/Off**

Scale/DIV

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision
DIV1|DIV2|DIV5|DIV10
:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision?
```

This command sets the per-division display scaling for the y-axis when scale type of Y axis is set to Log.

***RST:** 10 dB

Range: 1, 2, 5 or 10

Example:
```
:DISPlay:WINDow:TRACe:Y:PDIVision DIV5
```

Key access: **Amptd > Scale/DIV**

Scale Type

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing LIN-
ear|LOGarithmic
:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing?
```

Toggles the vertical graticule divisions between logarithmic unit and linear unit. The default logarithmic unit is dBm, and the linear unit is mV.

***RST:** Log

Example:
```
:DISPlay:WINDow:TRACe:Y:SPACing LINear
```

Key access: **Amptd > Scale Type > Log/Lin**

Power Units of Measure (Y Axis Unit)

```
:UNIT:POWer DBM|DBMV|DBMVEMF|DBUV|DBU-
VEMF|V|W|VEMF
:UNIT:POWer?
```

Specifies amplitude units for the input, output and display.

***RST:** dBm in log amplitude scale
Volts in linear amplitude scale

Example:
```
:UNIT:POWer DBMV
```

Key access: **Amptd > Y Axis Unit**

Hi-Sensitive

```
[:SENSe]:POWer[:RF]:HSENsitive[:STATe] OFF|ON|0|1
[:SENSe]:POWer[:RF]:HSENsitive [:STATe]?
```

Toggles the Hi-sensitive function between on and off.

Key access: **AMPTD > More > Hi-sensitivity**

Reference Offset Level

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet
<value>
:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet?
```

This command sets the amplitude offset level for the Y-Axis.

***RST:** 0 dB

Range: –327.60 to +327.60 dB

Example:
```
:DISPlay:WINDow:TRACe:Y:RLEVel:OFFSet 20dB
```

Key access: **Amptd > More > Ref Offset**

Correction Off

```
[:SENSe]:CORRection:OFF
```

Turn off the amplitude correction function off.

Key access: **Amptd > More > Amp Correction > Off**

Correction Apply State

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

Turns On or Off the amplitude corrections. When turned On, only the correction sets that were turned on are enabled. When turned Off, all of the correction sets are disabled. If there is no correction enabled, state can not be set to on

Example: `:CORRection:CSET:ALL ON`

Key access: **Amptd > Corrections > Apply Corrections > On**

Set Correction X state Off

```
[:SENSe]:CORRection:CSET[1]|2|3|4:OFF
[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe]?
```

Turns the amplitude correction function on/off.

Example: `:CORRection:CSET2:OFF`

Key access: **Amptd > Corrections > Correction1 > On**

Correction Antenna Unit State

```
[:SENSe]:CORRection:CSET:ANTUnit OFF|ON|0|1
[:SENSe]:CORRection:CSET:ANTUnit?
```

Turns the antenna unit on/off.

Example: `:CORRection:CSET:ANTUnit ON`

Key access: **Amptd > Corrections > Antenna Unit > On**

Read Correction X Data

```
[:SENSe]:CORRection:CSET[1]|2|3|4:DATA?
```

Query amplitude correction data.

Example: `:CORRection:CSET2:DATA?`

Current Correction Select

```
[:SENSe]:CORRection:SELect COR1|COR2|COR3|COR4
[:SENSe]:CORRection:SELect?
```

Set current correction for load COR file onto proper CorrectionX.

Example: `:CORRection:SELect?`

Input Impedance selection

```
[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]
OHM50|OHM75
[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]?
```

Toggles the input impedance correction between 50 Ω and 75 Ω in spectrum analyzer.

***RST:** OHM50

Example: `:CORRection:IMPedance OHM75`

Key access: **Amptd > More > Impedance**

PSD Mode State

```
:DISPlay:WINDow:TRACe:Y:PSD OFF|ON|0|1
:DISPlay:WINDow:TRACe:Y:PSD?
```

Use this command to turn on/off the power spectral density display mode.

Example: `:DISPlay:WINDow:TRACe:Y:PSD 1`

Key access: **Amptd > More > PSD Function > On/Off**

Bandwidth Subsection

Resolution Bandwidth

```
[:SENSe]:BANDwidth|BWIDth[:RESolution] <freq>
[:SENSe]:BANDwidth|BWIDth[:RESolution]?
```

Specifies the resolution bandwidth. For numeric entries, all RBW types choose the nearest (arithmetically, on a linear scale, rounding up) available RBW to the value entered.

Range: 10 Hz to 3 MHz

Default Unit: Hz

Example: `:BAND 1 kHz`

Key access: **BW > RBW**

Auto Resolution Bandwidth State

```
[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO
OFF|ON|0|1
[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO?
```

This command turns on/off auto resolution bandwidth state.

***RST:** On

Example: `:BWID:AUTO On`

Video Bandwidth

```
[:SENSe]:BANDwidth|BWIDth:VIDeo <freq>
[:SENSe]:BANDwidth|BWIDth:VIDeo?
```

Specifies the video bandwidth.

***RST:** 3 MHz

Range: 1 Hz to 3 MHz. This range is dependent upon the setting of `[:SENSe]:BANDwidth|BWIDth[:RESolution]`.

Default Unit: Hz

Example: `:BANDwidth:VIDeo 10 KHZ`

Key access: **BW > VBW Auto Man**

Auto Video Bandwidth State

```
[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO OFF|ON|0|1
[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO?
```

This command turns on/off auto video bandwidth state.

***RST:** On

Key access: **BW > VBW Auto/Man**

Video to Resolution Bandwidth Ratio

```
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio <number>
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio?
```

Specifies the ratio of the video bandwidth to the resolution bandwidth.

***RST:** 1.0

Range: 0.00001 to 3.0e6

Example:

```
:BANDwidth:VIDeo:RATio 30
```

Key access: **BW > VBW/RBW Ratio**

Auto Video to Resolution Bandwidth Ratio State

```
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio:AUTO
OFF|ON|0|1
[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio:AUTO?
```

This command turns on/off auto video to resolution bandwidth ratio.

***RST:** Auto

Example:

```
:BANDwidth:VIDeo:RATio:AUTO ON
```

Key access: **BW > VBW/RBW Auto Man**

Trace Subsection

Select Trace Display Mode

```
:TRACe1|2|3|4:MODE WRITe|MAXHold|MIN-
Hold|VIEW|BLANk
:TRACe1|2|3|4:MODE?
```

Selects the display mode for the selected trace.

WRITe puts the trace in the normal mode, updating the data.

MAXHold displays the highest measured trace value for all the data that has been measured since the function was turned on.

MINHold displays the lowest measured trace value for all the data that has been measured since the function was turned on.

VIEW turns on the trace data so that it can be viewed on the display.

BLANK turns off the trace data so that it is not viewed on the display.

Example: `:TRAC1:MODE VIEW`

Key access: **Trace > Clear Write | Max Hold | Min Hold | View | Blank**

Trace Max/Min Hold By Count

```
:TRACe[1]|2|3|4:HOLD:COUNt <value>
:TRACe[1]|2|3|4:HOLD:COUNt?
```

Use this command to set the Max/Min hold counter.

Example: `:TRACe:HOLD:COUNt 20`

Key access: **Trace > More > Max/Min Number On**

Add Trace

```
:TRACe:MATH:ADD
<destination_trace>,<source_trace1>,<source_trace2>
```

This command adds the selected trace on the activated trace.

Example: `:TRAC:MATH:ADD TRACE2,TRACE1,TRACE3`

Subtract Trace

```
:TRACe:MATH:SUBTract
<destination_trace>,<source_trace1>,<source_trace2>
```

This command subtract the selected trace from the activated trace.

Example: `:TRAC:MATH:SUBT TRACE2,TRACE1,TRACE3`

Query Trace Data

```
:TRACe[:DATA]? TRACe1|TRACe2|TRACe3|TRACe4|
```

This query command returns the current displayed data.

Example: `:TRACe:DATA? TRACE1`

Trace Math By

```
:TRACe:MATH:TYPE LOGPwr|POWer
:TRACe:MATH:TYPE?
```

This command toggles the trace math by log power and power.

Example: `:TRACe:MATH:TYPE LOGP`

Key access: **Trace > More > Trace Math By**

Trace Math Off

```
:TRACe:MATH:OFF
```

This command turns off the trace math function.

Key access: **Trace > More > Math Type > Off**

Trace Format

```
:FORMat[:TRACe][:DATA]ASCii|REAL
```

This command toggles the return trace data format between ASCii and real format.
ASC is standard ASCII string be separated by comma.
REAL is 4 bytes length float without comma.

Detector Subsection

Type of Detection

```
[:SENSe]:DETector:TRACe[1]|2|3|4[:FUNCtion]NEGa-
tive|POSitive|SAMPle|AVERage|NORMAL
[:SENSe]:DETector:TRACe[1]|2|3|4[:FUNCtion]?
```

Specifies the detection mode. For each trace interval (bucket), average detection displays the average of all the samples within the interval.

- **Negative peak** detection displays the lowest sample taken during the interval being displayed.
- **Positive peak** detection displays the highest sample taken during the interval being displayed.
- **Sample** detection displays the sample taken during the interval being displayed, and is used primarily to display noise or noise-like signals. In sample mode, the instantaneous signal value at the present display point is placed into memory. This detection should not be used to make the most accurate amplitude measurement of non noise-like signals.
- **Average** detection is used when measuring the average value of the amplitude across each trace interval (bucket). The averaging method used by the average detector is set to either video or power as appropriate when the average type is auto coupled.
- **Normal** detection selects the maximum and minimum video signal values alternately. When selecting **Normal** detection, "Norm" appears in the upper-left corner.

***PST:** Positive

Example: `:DETector:TRACe1 SAMPle`

Key access: **Trace > More > Detector**

Average Subsection

Average Type

```
[:SENSe]:AVERage:TYPE LOGPower|POWer|VOLTage
[:SENSe]:AVERage:TYPE?
```

Toggle the average type between Log power, power and voltage.

Example: `:AVERage:TYPE POW`

Key access: **BW > Average Type**

Average Number On/Off

```
[:SENSe]:AVERage:TRACe[1]|2|3|4:COUNt <integer>
[:SENSe]:AVERage:TRACe[1]|2|3|4:COUNt?
```

Specifies the number of measurements that are combined.

Range: 1 to 8192

```
[:SENSe]:AVERage:TRACe[1]|2|3|4[:STATe]
OFF|ON|0|1
[:SENSe]:AVERage:TRACe[1]|2|3|4[:STATe]?
```

This command toggles averaging off and on.

Example: `:AVERage:TRACe1:COUNt 20`

Key access: **Trace > More > Average**

Average Duration On/Off

```
[:SENSe]:AVERage:TRACe[1]|2|3|4:DURation <value>
[:SENSe]:AVERage:TRACe[1]|2|3|4:DURation?
```

Sets the duration of average process.

```
[:SENSe]:AVERage:TRACe[1]|2|3|4:DURation:STATe
OFF|ON|0|1
[:SENSe]:AVERage:TRACe[1]|2|3|4:DURation:STATe?
```

This command toggles the averaging duration off and on.

Example: `:AVERage:TRACe1:DURation 2S`

Key access: **Trace > More > Average Duration On/Off**

Average Restart

```
[:SENSe]:AVERage:TRACe[1]|2|3|4:CLEar
```

Restarts the trace average. This command is only available when average is on.

Example: `:AVERage:TRACe1:CLEar`

Key access: **Trace > More > Average Restart**

Sweep Subsection

Sweep Time

```
[:SENSe]:SWEep:TIME <time>
[:SENSe]:SWEep:TIME?
```

Specifies the time in which the instrument sweeps the display. A span value of 0 Hz causes the analyzer to enter zero span mode. In zero span the X-axis represents time rather than frequency.

```
[:SENSe]:SWEep:TIME:AUTO OFF|ON|0|1
[:SENSe]:SWEep:TIME:AUTO?
```

This command turns on/off auto sweep time state.

Example: `:SWEep:TIME 5s`

Key access: **Shift > Sweep > Sweep Time > Auto Man**

Sweep Speed

```
[:SENSe]:SWEep:SPEed NORMal|FAST
[:SENSe]:SWEep:SPEed?
```

Toggles the sweep speed between normal and fast.

***RST:** Fast

Example: `:SWEep:SPEed NORMal`

Key access: **Shift > Sweep > Sweep Setup > Sweep Speed Accuracy/Speed**

Sweep Mode

```
[:SENSe]:SWEep:MODE AUTO|FFT|SWEep
[:SENSe]:SWEep:MODE?
```

Sets the sweep mode manually.

***RST:** AUTO

Key access: **Shift > Sweep > Sweep Setup > Sweep Type**

Gated Sweep State

```
[:SENSe]:SWEep:EGATe[:STATe] OFF|ON|0|1
[:SENSe]:SWEep:EGATe[:STATe]?
```

Use this command to turn on/off gated sweep mode.

Example:
```
:SWEep:EGATe ON
```

Key access: **Shift > Sweep > Gated Sweep > Gated Sweep > On/Off**

Gate View

```
[:SENSe]:SWEep:EGATe:VIEW ON|OFF|1|0
[:SENSe]:SWEep:EGATe:VIEW?
```

Use this command to turn on/off gate view. The gated view can be enable when the gated sweep is turned on.

Example:
```
:SWEep:EGATe:VIEW ON
```

Key access: **Shift > Sweep > Sweep Setup > Gated Sweep > Gate View > On**

Gate View Sweep Time

```
[:SENSe]:SWEep:EGATe:TIME <time>
[:SENSe]:SWEep:EGATe:TIME?
```

Set the gated view sweep time. The time unit of this command is second.

Example:
```
:SWEep:EGATe:TIME 0.002
```

Key access: **Shift > Sweep > Sweep Setup > Gated Sweep > View Sweep Time**

Gate Delay

```
[:SENSe]:SWEep:EGATe:DELay <time>
[:SENSe]:SWEep:EGATe:DELay?
```

Set the gate delay time.

Example: `:SWEep:EGATe:DELay 0.003`

Key access: **Shift > Sweep > Sweep Setup > Gated Sweep > Gate Delay**

Gate Length

```
[:SENSe]:SWEep:EGATe:LENGth <time>
[:SENSe]:SWEep:EGATe:LENGth?
```

Set the gate length of the gated sweep.

Example: `:SWEep:EGATe:LENGth 0.5`

Key access: **Shift > Sweep > Sweep Setup > Gated Sweep > Gate Length**

Gate Source

```
[:SENSe]:SWEep:EGATe:SOURce EXTer-
nal|FRAMe|RFBurst
[:SENSe]:SWEep:EGATe:SOURce?
```

Set the gate trigger source of gate sweep.

Example: `:SWEep:EGATe:SOURce?`

Key access: **Shift > Sweep > Sweep Setup > Gated Sweep > Gate Delay**

Display Subsection

Graticule State

```
:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe]
OFF|ON|0|1
:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe]?
```

This command toggles the graticule between on and off.

Example: `:DISPlay:WINDow:TRACe:GRATicule:GRID OFF`

Key access: **Shift > Disp > Graticule On/Off**

Y Scale State

```
:DISPlay:WINDow:TRACe:YSCale[:STATe] OFF|ON|0|1
:DISPlay:WINDow:TRACe:YSCale[:STATe]?
```

This command toggles the Y Scale between on and off.

Example: `:DISPlay:WINDow:TRACe:YSCale ON`

Key access: **Shift > Disp > Y Scale On/Off**

Display Line

```
:DISPlay:WINDow:TRACe:Y:DLINe <value>
:DISPlay:WINDow:TRACe:Y:DLINe?
```

Sets the amplitude value for the display line.

```
:DISPlay:WINDow:TRACe:Y:DLINe:STATe OFF|ON|0|1
:DISPlay:WINDow:TRACe:Y:DLINe:STATe?
```

Toggles the display line between on and off.

Example: `:DISPlay:WINDow:TRACe:Y:DLINe -10`

Key access: **Shift > Disp > Display LIne**

Calculate Subsystem

This subsystem is used to perform post-acquisition data processing. In effect, the collection of new data triggers the CALCulate subsystem. In this instrument, the primary functions in this subsystem are markers and limits.

Limit Line Subsection

Limit lines can be defined for your measurement. You can then have the instrument compare the data to your defined limits and indicate a pass/fail condition.

Type of Limit Line

```
:CALCulate:LLINe[1]:TYPE UPPer|LOWer
:CALCulate:LLINe[1]:TYPE?
```

Sets a limit line to be either an upper or lower type limit line. An upper line will be used as the maximum allowable value when comparing with the data.

Example: `:CALCulate:LLINe:TYPE UPPer`

Key access: **Shift > Limit > Limit 1 > Limit Type**

Limit Line State

```
:CALCulate:LLINe[1]|2:STATe OFF|ON|0|1
:CALCulate:LLINe[1]|2:STATe?
```

Toggles the limit line function between on and off.

Key access: **Shift > Limit > Limit 1> Limit Line On/Off**

Fixed/Relative Limit

```
:CALCulate:LLINe:CMODe FIXed|RELative
:CALCulate:LLINe:CMODe?
```

Toggles the limit line mode between fixed and relative.

Key access: **Shift > Limit > Limits Fixed/Rel**

Limit Line Y-axis Value

```
:CALCulate:LLINe[1]:Y <value>
:CALCulate:LLINe[1]:Y?
```

Sets the Y-axis value of a limit line. Limit line Y-axis value is set independently and is not affected by the X-axis units.

***RST:** 0 dBm

Example:
```
:CALCulate:LLINe:Y -20dBm
```

Key access: **Shift > Limit> Limit 1 > Limit Line**

Limit Line X-axis Value

```
:CALCulate:LLINe:CONTrol:DOMain FREQuency|TIME
:CALCulate:LLINe:CONTrol:DOMain?
```

Toggles the limit line X-axis value between frequency and time.

Example:
```
:CALCulate:LLINe:CONTrol:DOMain FREQuency
```

Key access: **Shift > Limit > X Axis Units**

Limit State

```
:CALCulate:LLINe[1]|2:DISPlay OFF|ON|0|1
:CALCulate:LLINe[1]|2:DISPlay?
```

Toggles the limits state between on and off.

Key access: **Shift > Limit > Limit 1> Limit On/Off**

Query Current Limits Result

```
:CALCulate:LLINe[1]|2:FAIL?
```

This query returns the limits pass/failed result. If the test result fail, this command will get result 1. If the test result pass the limit, it will get result 0.

Example:
```
:CALCulate:LLINe:FAIL?
```

Limit Beep State

```
:CALCulate:LLINe:CONTrol:BEEP OFF|ON|0|1
:CALCulate:LLINe:CONTrol:BEEP?
```

Use this command to turn on/off the limit beep status.

Example: `:CALCulate:LLINe:CONTrol:BEEP ON`

Key access: **Shift > Limit > X Axis Units**

Define Limits Values

```
:CALCulate:LLINe[1]|2:DATA
<x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<co
nnected>}
:CALCulate:LLINe[1]|2:DATA?
```

Use this command to define the limits values.

Example: `:CALC:LLIN1:DATA 10000000,-20,0,20000000,-30,1`

Key access: **Shift > Limit > Limit 1 > Edit Limits**

Delete Limits Values

```
:CALCulate:LLINe[1]|2:DELete
```

Use this command to delete the limits values.

Example: `:CALCulate:LLINe:DELete`

Delete All Limits Data

```
:CALCulate:LLINe:ALL:DELete
```

Use this command to define all the limits values.

Example: `:CALCulate:LLINe:ALL:DELete`

Query Limits Result

```
:CALCulate:LLINe[1]|2:FAIL?
```

Use this command to query the limit result.

Example: `:CALCulate:LLINe1:FAIL?`

Set the limits Margin Value

```
:CALCulate:LLINe[1]|2:MARGin <value>
:CALCulate:LLINe[1]|2:MARGin?
```

Use this command to define the limits margin values.

Example: `:CALCulate:LLINe1:MARGin 20`

Key access: **Shift > Limit > Limit 1 > More > Margin**

Set Limits Margin State

```
:CALCulate:LLINe[1]|2:MARGin:STATe OFF|ON|0|1
:CALCulate:LLINe[1]|2:MARGin:STATe?
```

Use this command to turn on/off limit margin.

Example: `:CALCulate:LLINe:MARGin:STATe 1`

Key access: **Shift > Limit > Limit 1 > More > Margin > On**

Limits Statistics Threshold State

```
:CALCulate:LLINe[1]|2:STHReshold:STATe
OFF|ON|0|1
:CALCulate:LLINe[1]|2:STHReshold:STATe?
```

Use this command to turn on/off the limit threshold.

Example: `:CALCulate:LLINe:STHReshold:STATe 1`

Key access: **Shift > Limit > Limit 1 > More > Margin > Stat. Threshold**

Limits Statistics Threshold Value

```
:CALCulate:LLINe[1]|2:STHReshold <val>
:CALCulate:LLINe[1]|2:STHReshold?
```

Use this command to set and query the limits statistics threshold value.

Example: `:CALCulate:LLINe:STHReshold 10`

Key access: **Shift > Limit > Limit 1 > More > Margin > Stat. Threshold**

Marker Subsection

Markers All Off on All Traces

```
:CALCulate:MARKer:AOFF
```

Turns off all markers on all the traces.

Key access: **Marker > All Off**

Continuous Peaking Marker

```
:CALCulate:MARKer[1]|2|3|4|5|6:CPEak[:STATe]
OFF|ON|0|1
:CALCulate:MARKer[1]|2|3|4|5|6:CPEak[:STATe]?
```

Toggles the continuous peak search function between on and off.

Example:
```
:CALCulate:MARKer1:CPEak ON
```

Key access: **Shift > Peak > More > Continuous PK**

Frequency Counter Marker

```
:CALCulate:MARKer[1]|2|3|4|5|6:FCOunt[:STATe]
OFF|ON|0|1
:CALCulate:MARKer[1]|2|3|4|5|6:FCOunt[:STATe]?
```

Turns on/off the marker frequency counter. To query the frequency counter, use `:CALCulate:MARKer[1]:FCOunt:X?` If the specified marker number is not the active marker, it becomes the active marker. If the specified marker number is not on, it is turned on and becomes the active marker. A 1 is returned only if marker count is on and the selected number is the active marker.

***RST:** Off

Remarks: If a frequency count x value is generated when the frequency count state is off, then 0 is returned.

Example:
```
:CALCulate:MARKer1:FCOunt 1
```

Key access: **Marker > Function > Counter**

Marker Function

```
:CALCulate:MARKer[1]|2|3|4|5|6:FUNCtion
FCOunt|NOISe|OFF
:CALCulate:MARKer[1]|2|3|4|5|6:FUNCtion?
```

This command selects the marker function for the designated marker.

`COunt` refers to the frequency counter function.

`NOISe` refers to the noise measurement function.

`OFF` refers to the normal function.

***RST:** 1 kHz

Example: `:CALCulate:MARKer1:FUNCtion NOISe`

Key access: **Marker > Function**

Peak Search

```
:CALCulate:MARKer[1]|2|3|4|5|6:MAXimum
```

Performs a peak search based on the search mode settings of `:CALCulate:MARKer[1]|2|3|4|5|6:FUNCtion FCOunt|NOISe|OFF` and `:CALCulate:MARKer:PEAK:SEARch:MODE MAXimum|MINimum`.

Example: `:CALCulate:MARKer:PEAK:SEARch:MODE MAXimum`

Key access: **Shift > Peak > Peak Search**

Next Peak Search

```
:CALCulate:MARKer[1]|2|3|4|5|6:MAXimum:NEXT
```

Places the selected marker on the next highest signal peak of the current marked peak.

Example: `:CALCulate:MARKer1:MAXimum:NEXT`

Key access: **Shift > Peak > Next Peak**

Marker Peak Left/Right Search

```
:CALCulate:MARKer[1]|2|3|4|5|6:MAXimum:LEFT
:CALCulate:MARKer[1]|2|3|4|5|6:MAXimum:RIGHt
```

Places the selected marker on the next highest signal peak to the left/right of the current marked peak.

Example: `:CALCulate:MARKer1:MAXimum:LEFT`

Key access: **Shift >Peak > Next Left\Right Peak**

Marker Mode

```
:CALCulate:MARKer[1]|2|3|4|5|6:MODE POSi-
tion|DELTa|OFF
:CALCulate:MARKer[1]|2|3|4|5|6:MODE?
```

Selects the type of markers that you want to activate.

`Position` selects a normal marker that can be positioned on a trace and from which trace information will be generated.

`Delta` activates a pair of markers, one of which is fixed at the current marker location. The other marker can then be moved around on the trace. The marker readout shows the difference between the two markers.

`Off` turns the designated marker off. If a marker is not active when the mode is queried, "`Off`" will be returned.

Example: `:CALCulate:MARKer1:MODE DELTa`

Key access: **Marker > Mode > Normal/Delta/Off**

Peak Excursion

```
:CALCulate:MARKer:PEAK:EXCursion <value>
:CALCulate:MARKer:PEAK:EXCursion?
```

Specifies the minimum signal excursion above the threshold for the internal peak identification routine to recognize a signal as a peak.

Example: `:CALCulate:MARKer:PEAK:EXCursion 6 dB`

Key access: **Shift > Peak > More > PK Criteria > Peak Excursion**

Peak Search Type

```
:CALCulate:MARKer:PEAK:SEARch:MODE MAXimum|MINimum
:CALCulate:MARKer:PEAK:SEARch:MODE?
```

Specifies the peak search type. When it is set to maximum, the places the selected marker on the highest point on the assigned trace. When it is set to minimum, the places the selected marker on the lowest point on the assigned trace.

Example: `:CALCulate:MARKer:PEAK:SEARch:MODE MINimum`

Key access: **Shift > Peak > More > PK Criteria > Peak Type**

Peak Threshold

```
:CALCulate:MARKer:PEAK:THReshold <value>
:CALCulate:MARKer:PEAK:THReshold?
```

Specifies the minimum signal level for the analyzers internal peak identification routine to recognize a signal as a peak. This applies to all traces and all windows.

Preset: –140 dBm

Example: `:CALCulate:MARKer:PEAK:THReshold -30 dBm`

Key access: **Shift > Peak > More > PK Criterion > Peak Threshold**

Peak to Peak Delta Markers

```
:CALCulate:MARKer[1]|2|3|4|5|6:PTPeak
```

Positions a pair of delta markers on the highest and lowest points on the trace.

Example: `:CALCulate:MARKer2:PTPeak`

Key access: **Peak > Pk-Pk Search**

Marker to Center

```
:CALCulate:MARKer[1]|2|3|4|5|6[:SET]:CENTer
```

This command sets the center frequency equal to the specified marker frequency, which moves the marker to the center of the screen. In delta marker mode, the center frequency is set to the marker delta value. This command is not available in zero span. This command is just available in Spectrum Analyzer and Tracking Generator mode.

Example: `:CALCulate:MARKer1:CENTer`

Key access: **Marker > Marker To > To Center**

Marker to Start Frequency

```
:CALCulate:MARKer[1]|2|3|4|5|6[:SET]:STARt
```

Sets the start frequency to the value of the specified marker frequency. In delta marker mode, the start frequency is set to the marker delta value. This command is not available in zero span. This command is just available in Spectrum Analyzer and Tracking Generator mode.

Example: `:CALCulate:MARKer:STARt`

Key access: **Marker > Marker To > To Start**

Marker to Stop Frequency

```
:CALCulate:MARKer[1]|2|3|4|5|6[:SET]:STOP
```

Sets the stop frequency to the value of the specified marker frequency. In delta marker mode, the stop frequency is set to the marker delta value. This command is not available in zero span. This command is just available in Spectrum Analyzer and Tracking Generator mode.

Example: `:CALCulate:MARKer:STOP`

Key access: **Marker > Marker To > To Stop**

Marker On/Off

```
:CALCulate:MARKer[1]|2|3|4|5|6:STATe OFF|ON|0|1
:CALCulate:MARKer[1]|2|3|4|5|6:STATe?
```

This command toggles the selected marker status between on and off.

Example: `:CALCulate:MARKer1:STATe 0`

Key access: **Marker > Mode > Normal/Off**

Marker to Trace

```
:CALCulate:MARKer[1]|2|3|4|5|6:TRACe <integer>
:CALCulate:MARKer[1]|2|3|4|5|6:TRACe?
```

This command assigns the specified marker to the designated trace 1, 2, 3 or 4.

Example: `:CALCulate:MARKer2:TRACe 2`

Key access: **Marker > Marker Trace**

Marker X value

```
:CALCulate:MARKer[1]|2|3|4|5|6:X <para>
:CALCulate:MARKer[1]|2|3|4|5|6:X?
```

This command positions the designated marker on its assigned trace at the specified trace X value.

The value is in the X-axis units, which can be a frequency or time.

The query returns the current X value of the designated marker.

Example: `:CALCulate:MARKer2:X 2000 MHz`

Key access: **Marker > Normal**

Marker Readout X Value

```
:CALCulate:MARKer[1]|2|3|4|5|6:X:READout
FREQuency|TIME |PERiod
:CALCulate:MARKer[1]|2|3|4|5|6:X:READout?
```

Toggles the marker X-Axis readout between frequency, time and period.

Key access: **Marker > More > Read Out**

Query Marker Readout: Y Value

```
:CALCulate:MARKer[1]|2|3|4|5|6:Y?
```

This command reads the current Y value for the designated marker or delta on its assigned trace. The value is in the Y-axis units for the current trace.

This command can be used to read the results of marker functions such as and noise that are displayed in the marker value field on the analyzer.

Example: `:CALCulate:MARKer1:Y?`

Marker Demod Type

```
:CALCulate:MARKer:DEMod:TYPE AM|FM
:CALCulate:MARKer:DEMod:TYPE?
```

Toggles the marker demodulation function between amplitude modulation and frequency modulation.

Key access: **Marker > Mode > Demod Setting > Demod Type**

Marker Demod Speaker Volume

```
:CALCulate:MARKer:DEMod:SVOLume <para>
:CALCulate:MARKer:DEMod:SVOLume?
```

Sets the speaker volume for the demodulated signal.

Key access: **Marker > Mode > Demod Setting > Speaker Vol**

Marker Demod Delay Time

```
:CALCulate:MARKer:DEMod:DTIMe <para>
:CALCulate:MARKer:DEMod:DTIMe?
```

Sets the delay time for the marker demodulation.

Key access: **Marker > Mode > Demod Setting > Delay Time**

Initiate Subsystem

The INITiate subsystem is used to control the initiation of the spectrum analyzer sweep operation.

Continuous or Single Sweep

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

Selects whether the sweep operation is continuously initiated or not.

This command affects sweep if not in a measurement, and affects trigger when in a measurement. This corresponds to continuous sweep or single sweep operation when not in a measurement, and continuous measurement or single measurement operation when in a measurement.

When not in a measurement, this command does the following:

- When ON at the completion of each sweep cycle, the sweep system immediately initiates another sweep cycle.
- When OFF, the sweep system remains in an "idle" state until `CONTinuous` is ON or `:INITiate[:IMMediate]` is received. On receiving the `:INITiate[:IMMediate]` command, it will go through a single sweep cycle, and then return to the "idle" state.
- The query returns 1 or 0 into the output buffer. 1 is returned when there is continuous sweeping. 0 is returned when there is only a single sweep.

When in a measurement, this command does the following:

- When ON at the completion of each trigger cycle, the trigger system immediately initiates another trigger cycle.
- When OFF, the trigger system remains in an "idle" state until `CONTinuous` is ON or `:INITiate[:IMMediate]` is received. On receiving the `:INITiate[:IMMediate]` command, it will go through a single trigger cycle, and then return to the "idle" state.
- The query returns 1 or 0 into the output buffer. 1 is returned when in a continuous measurement state. 0 is returned when there is only a single measurement.

***RST:** Continuous

Key access: **Sweep > Sweep Single/Cont**

Initiate a Single Sweep

`:INITiate[:IMMediate]`

This command initiates a sweep if not in a measurement. If in a measurement, it triggers the measurement.

Remarks: See also the `*TRG` command

Use the `:TRIGer[:SEQuence]:SOURce EXTernal` command to select the external trigger.

The analyzer must be in the single measurement mode. If `:INITiate:CONTinuous` is ON, the command is ignored.

If the analyzer is in signal identification mode, two sweeps are required, as this mode relies on the acquisition of data from two successive sweeps. Therefore, if the analyzer is in single sweep mode, two sweep triggers are needed to generate the sweep pair. In image suppress mode, synchronization is ensured by first turning off signal identification, initiating a single sweep, then turning on signal identification followed by two single sweeps.

Example: `:INITiate`

Key access: **Sweep > Single Sweep**

Trigger Subsystem

The TRIGger subsystem is used to set the controls and parameters associated with triggering the data acquisitions. This subsystem is only valid when the analyzer is in zero span.

Trigger Source

```
:TRIGger[:SEQuence]:SOURce IMMediate|VIDeo|EXTer-
nal|RFBurst
:TRIGger[:SEQuence]:SOURce?
```

Specifies the source (or type) of triggering used to start a measurement.

Immediate is free-run triggering

Video triggers on the video signal level

External allows you to connect an external trigger source

RF Burst triggers on the RF signal level.

***RST:** Immediate (free-run triggering)

Key access: **Shift > Sweep > Trigger > Free Run | Video | External | RFBurst**

Trigger Delay Time State

```
:TRIGger[:SEQuence]:DELaytime:STATe OFF|ON|0|1
:TRIGger[:SEQuence]:DELaytime:STATe?
```

Use this command to turn on/off trigger delay.

Example: :TRIGger:DELaytime:STATe OFF

Trigger Delay Time

```
:TRIGger[:SEQuence]:DELaytime <time>
:TRIGger[:SEQuence]:DELaytime?
```

Specifies trigger delay time for video or external trigger signal.

***RST:** 6 µs

Example: `:TRIGger:DELaytime 10 us`

Key access: **Shift > Sweep > Trigger >Video >Trig Delay**

Video Trigger Level Amplitude

```
:TRIGger[:SEQuence]:VIDeo:LEVel <value>
:TRIGger[:SEQuence]:VIDeo:LEVel?
```

Specifies the level at which a video trigger will occur. Video is adjusted using this command, but must also be selected using the command `:TRIGger[:SEQuence]:SOURce VIDeo`.

Range: 10 display divisions below reference level to reference level

Default Unit: current amplitude units

Example: `:TRIGger:VIDeo:LEVel -20 dBm`

Key access: **Shift > Sweep > Trigger > Video >Video Level**

External Trigger Slope

```
:TRIGger[:SEQuence]:EXTernal:SLOPe POSitive|NEGative
:TRIGger[:SEQuence]:EXTernal:SLOPe?
```

This command activates the trigger condition that allows the next sweep to start when the external voltage (connected to **EXT TRIG IN** connector) passes through approximately 1.5 volts. The external trigger signal must be a 0 V to +5 V TTL signal. This function only controls the trigger polarity (for positive or negative-going signals).

Before you set the trigger slope, The external trigger source must be selected using the command `:TRIGger[:SEQuence]:SOURce EXTernal`.

***RST:** Positive

Key access: **Shift > Sweep > Trigger > Trigger Slop**

RF Burst Level

```
:TRIGger[:SEQuence]:RFBurst:LEVel <value>
:TRIGger[:SEQuence]:RFBurst:LEVel?
```

Specifies the level at which a RF burst trigger will occur. Before you set the RF burst trigger level, the RF burst trigger must be selected using the command `:TRIGger[:SEQuence]:SOURce RFBurst.`

Example: `:TRIGger:RFBurst:LEVel -30`

Key access: **Shift > Sweep > Trigger > RFBurst > RFBurst Level**

RF Burst Trigger Slope

```
:TRIGger[:SEQuence]:RFBurst:SLOPe POSitive|NEGative
:TRIGger[:SEQuence]:RFBurst:SLOPe?
```

This command activates the trigger condition that allows the next sweep to start when the external RF signal (connected to **RF IN** connector) passes RF burst trigger level.

Before you set the trigger slope, The external trigger source must be selected using the command `:TRIGger[:SEQuence]:SOURce RF Burst`.

***RST:** Positive

Key access: **Shift > Sweep > Trigger > RF Burst > Trigger Slop**

Periodic Timer Period

```
:TRIGger[:SEQuence]:FRAMe:PERiod <time>
:TRIGger[:SEQuence]:FRAMe:PERiod?
```

Specifies periodic timer period under gate sweep mode.

Example: `:TRIGger:FRAMe:PERiod 10ms`

Key access: **Shift > Sweep > Gated Sweep > Gate Source >Periodic Timer, then Timer Setup > Period**

Periodic Timer Offset

```
:TRIGger[:SEQuence]:FRAMe:OFFSet <time>
:TRIGger[:SEQuence]:FRAMe:OFFSet?
```

Specifies periodic timer offset under gate sweep mode.

Example: `:TRIGger:FRAMe:OFFSet 10ms`

Key access: **Shift > Sweep > Gated Sweep > Gate Source >Periodic Timer, then Timer Setup > Offset**

Synchronization Source

```
:TRIGger[:SEQuence]:FRAMe:SYNC OFF|EXTernal
|RFBurst
:TRIGger[:SEQuence]:FRAMe:SYNC?
```

Specifies the source of period timer under gate sweep mode.

Example: `:TRIGger:FRAMe:SYNC OFF`

Key access: **Shift > Sweep > Gated Sweep > Gate Source >Periodic Timer, then Timer Setup > Sync Source**

Power Measurement Subsystem

The Power Measurement Subsystem provides you the SCPI commands reference for the N9322C built in one button power measurement function, such as Adjacent Channel Power Ratio (ACPR) , Channel Power (CHP) , Occupied Bandwidth (OBW) and Spectrum Emission Mask (SEM). Use the command `:INSTrument:MEASure OFF|CHPower|ACPR|OBW|SPECtrogram|SEM|CHScanner` to access different power measurement mode.

ACPR Subsection

Center Freq

`[:SENSe]:ACPRatio:FREQuency:CENTer <freq>`

`[:SENSe]:ACPRatio:FREQuency:CENTer?`

Sets the center frequency of the main channel power.

Range: 100.5 kHz to 6.9999995 GHz

Example: `:ACPRatio:FREQuency:CENTer 3.4GHz`

Key access: **Meas > Center Freq**

Main Channel

`[:SENSe]:ACPRatio:BANDwidth|BWIDth:INTegration <freq>`

`[:SENSe]:ACPRatio:BANDwidth|BWIDth:INTegration?`

Specifies the range of integration used in calculating the power in the main channel.

***RST:** 1 MHz

Range: 300 Hz to 25 MHz

Example: `:ACPRatio:BAND:INTegration 2 MHz`

Key access: **Meas > Main Channel**

Adjacent Channel State

```
[:SENSe]:ACPRatio:OFFSet:LIST:STATe
OFF|ON|0|1,OFF|ON|0|1,OFF|ON|0|1,OFF|ON|0|1,OFF|
ON|0|1,OFF|ON|0|1
```

```
[:SENSe]:ACPRatio:OFFSet:LIST:STATe?
```

Specifies the range of integration used in calculating the power in the adjacent channel.

***RST:** 1 MHz

Range: 300 Hz to 25 MHz

Example: `:ACPRatio:OFFSet:LIST:STATe 1,1,1,1,1,1`

Key access: **Meas > Adj Chn State**

Adjacent Channel Bandwidth

```
[:SENSe]:ACPRatio:OFFSet:LIST:BAND-
width|BWIDth[INTegration]
<freq>,<freq>,<freq>,<freq>,<freq>
```

```
[:SENSe]:ACPRatio:OFFSet:LIST:BAND-
width|BWIDth[INTegration]?
```

Specifies the bandwidth used in calculating the power in the adjacent channel.

```
[:SENSe]:ACPRatio:OFFSet:LIST:BAND-
width|BWIDth[INTegration]:AUTO
OFF|ON|0|1,OFF|ON|0|1,OFF|ON|0|1,OFF|ON|0|1,OFF|
ON|0|1,OFF|ON|0|1
```

```
[:SENSe]:ACPRatio:OFFSet:LIST:BANDwidth|BWIDth[I
NTegration]:AUTO?
```

Set the adjacent channel bandwidth to auto mode.

***RST:** 1 MHz

Range: 300 Hz to 25 MHz

Example: `:ACPR:OFFS:LIST:BAND 1e6,1e6,1e6,1e6,1e6,1e6`

Key access: **Meas > Adj Chn BW**

Channel Space

```
[:SENSe]:ACPRatio:OFFSet:LIST[FREQuency]
<freq>,<freq>,<freq>,<freq>,<freq>
```

```
[:SENSe]:ACPRatio:OFFSet:LIST[FREQuency]?
```

Sets the space value between the center frequency of main channel power and that of the adjacent channel power.

```
[:SENSe]:ACPRatio:OFFSet:LIST[FREQuency]:AUTO
OFF|ON|0|1,OFF|ON|0|1,OFF|ON|0|1,OFF|ON|0|1,OFF|
ON|0|1,OFF|ON|0|1
```

```
[:SENSe]:ACPRatio:OFFSet:LIST[FREQuency]ATUO?
```

Sets the space value between the center frequency of main channel power and that of the adjacent channel power to auto mode.

Range: 0 Hz to 25 MHz

Example: `:ACPR:OFFS:LIST 1e6,2e6,2e6,3e6,5e6,6e6`

Key access: **Meas > Adj Chn Space**

Query Main Channel Power Result

```
:MEASure:ACPRatio:MAIN?
```

Return the main channel power of ACPR measurement.

Query Lower Adjacent Channel Power

```
:MEASure:ACPRatio:LOWer:POWer?
```

Return the lower adjacent channel power of ACPR measurement.

Query Lower Adjacent Channel Power Ratio

```
:MEASure:ACPRatio:LOWer?
```

Return the lower adjacent channel power to main channel power ratio.

Query Upper Adjacent Channel Power

```
:MEASure:ACPRatio:UPPer:POWer?
```

Return the upper adjacent channel power of ACPR measurement.

Query Upper Adjacent Channel Power Ratio

```
:MEASure:ACPRatio:UPPer?
```

Return the upper adjacent channel power to main channel power ratio.

ACPR RRC Filter State

```
[:SENSe]:ACPRatio:RRC[:STATe] OFF|ON|0|1
[:SENSe]:ACPRatio:RRC[:STATe]?
```

Use this command to turn on/off the RRC filter.

Example: `:ACPRatio:RRC 1`

Key access: **Meas > RRC filter > On/Off**

ACPR RRC Filter Alpha Value

```
[:SENSe]:ACPRatio:RRC:ALPHa <val>
[:SENSe]:ACPRatio:RRC:ALPHa?
```

Use this command to set the alpha value of RRC filter.

Example: `:ACPRatio:RRC:ALPHa 0.35`

Key access: **Meas > RRC filter**

CHP Subsection

Center Freq

```
[:SENSe]:CHPower:FREQuency:CENTer <freq>
[:SENSe]:CHPower:FREQuency:CENTer?
```

Sets the center frequency of the display.

Example: `:CHPower:FREQuency:CENTer 4 GHz`

Key access: **Meas > Center Freq**

Integration BW

```
[:SENSe]:CHPower:BWIDth|BANDwidth:INTegration <freq>
[:SENSe]:CHPower:BWIDth|BANDwidth:INTegration?
```

Specifies the integration bandwidth to calculate the power.

Example: `:CHPower:BWIDth:INTegration 40 MHz`

Key access: **Meas > Integrated BW**

Channel Span

```
[:SENSe]:CHPower:FREQuency:SPAN <freq>
[:SENSe]:CHPower:FREQuency:SPAN?
```

Sets the analyzer span for the channel power measurement. Be sure the span is set between 1 and 10 times the integration bandwidth.

Example: `:CHPower:FREQuency:SPAN 50 MHz`

Key access: **Meas > Integrated BW**

Channel Span Power

```
[:SENSe]:CHPower:FREQuency:SPAN:POWer
```

Sets the integrated bandwidth equal to the Span.

Key access: **Meas >Span Power**

Channel Power Display Mode

```
[:SENSe]:CHPower:DISPlay CHARt|METer
[:SENSe]:CHPower:DISPlay?
```

Toggles the channel power display mode between chart and meter.

***RST:** chart

Example: `:CHPower:DISPlay METer`

Key access: **Meas >Disp Mode**

Channel Power Auto Range

```
[:SENSe]:CHPower:METer:RANGe:AUTO OFF|ON|0|1
[:SENSe]:CHPower:METer:RANGe:AUTO?
```

Toggles the channel power auto range between on and off. It's only available when the display mode is set to Meter.

***RST:** ON

Example:
```
:CHPower:METer:RANGe:AUTO 0
```

Key access: **Meas > Disp Range > Auto Range**

Channel Power Range Top

```
[:SENSe]:CHPower:METer:RANGe:TOP <value>
[:SENSe]:CHPower:METer:RANGe:TOP?
```

Set the top amplitude range of the channel power measurement. it's only available when the channel power auto range is set to off.

***RST:** –10

Range: Bottom to –10

Example:
```
:CHPower:METer:RANGe:TOP -30
```

Key access: **Meas >Disp Range > Top**

Channel Power Range Bottom

```
[:SENSe]:CHPower:METer:RANGe:BOTTom <value>
[:SENSe]:CHPower:METer:RANGe:TOP?
```

Set the bottom amplitude range of the channel power measurement. it's only available when the channel power auto range is set to off.

***RST:** –70

Range: –140 to Top

Example:
```
:CHPower:METer:RANGe:BOTTom -60
```

Key access: **Meas >Disp Range > Bottom**

Query Channel Power and Density Result

```
:MEASure:CHPower?
```

This command returns scalar results of main channel power, and power density.

```
:MEASure:CHPower:CHPower?
```

This command returns the value of the channel power in amplitude units.

```
:MEASure:CHPower:DENSity?
```

This command returns the value of the channel power density in amplitude units/Hz.

Channel Power RRC Filter State

```
[:SENSe]:CHPower:RRC[:STATe] OFF|ON|0|1
[:SENSe]:CHPower:RRC[:STATe]?
```

Use this command to turn on/off RRC filter in Channel Power test.

Example: `:CHPower:RRC 1`

Key access: **Meas > More > RRC Filter**

Channel Power RRC Filter Alpha Value

```
[:SENSe]:CHPower:RRC:ALPHa <val>
[:SENSe]:CHPower:RRC:ALPHa?
```

Use this command to set the Alpha value of RRC filter in channel power test.

Range: 0.01 to 1

Example: `:CHPower:RRC:ALPHa 0.5`

Key access: **Meas > More > RRC Filter**

Channel Power RRC Filter Bandwidth

```
[:SENSe]:CHPower:RRC:BW <value>
[:SENSe]:CHPower:RRC:BW?
```

Use this command to set the bandwidth of RRC filter in channel power test.

Example: `:CHPower:RRC:BW 2 MHZ`

Key access: **Meas > More > Filter BW**

OBW Subsection

Select the measurement method of OBW

```
[:SENSe]:OBW:METHod PERCent|DBC
[:SENSe]:OBW:METHod?
```

This command toggles the method of OBW measurement between percent and dBc.

***RST:** Percent

Example: `:CHPower:RRC:BW 2 MHZ`

Key access: **Meas > Occupied BW > Method**

Set percentage (%) method of OBW

```
[:SENSe]:OBW:PERCent <para>
[:SENSe]:OBW:PERCent?
```

edit the percentage of signal power used when determining the occupied bandwidth. Press **{%}** to set the percentage ranging from 10.00% to 99.99%.

***RST:** 99

Range: 10 to 99

Example: `:OBW:PERCent 22.32`

Key access: **Meas > Occupied BW > %**

Set dBc method of OBW

```
[:SENSe]:OBWidth:XDB <value>
[:SENSe]:OBWidth:XDB?
```

specify the power level used to determine the emission bandwidth as the number of dB down from the highest signal point, within the occupied bandwidth span.

***RST:** 26

Range: 0.10 to 100

Example: `:OBWidth:XDB 4`

Key access: **Meas > Occupied BW > dBc**

OBW Span

```
[:SENSe]:OBWidth:FREQuency:SPAN <freq>
[:SENSe]:OBWidth:FREQuency:SPAN?
```

Use this command to set the span in OBW test.

Example: `:OBWidth:FREQuency:SPAN 50 MHz`

Key access: **Span > Span**

OBW Max Hold State

```
[:SENSe]:OBWidth:MAXHold OFF|ON|0|1
[:SENSe]:OBWidth:MAXHold?
```

Use this command to turn on/off trace max hold state in OBW test.

Example: `:OBWidth:MAXHold 1`

Key access: **Trace > Max Hold**

Query OBW and Centroid Result

```
:MEASure:OBW?
:FETCh:OBW?
```

Use this command to query the occupied bandwidth and bandwidth centroid according to the method you set.

Query OBW Result

```
:MEASure:OBW:OBW?
:FETCh:OBW:OBW?
```

Use this command to query the occupied bandwidth according to the method you set.

Query Centroid Result

```
:MEASure:OBW:CENTroid?
:FETCh:OBW:CENTroid?
```

Use this command to query the occupied bandwidth according to the method you set.

SEM Subsection

SEM Average Number

```
[:SENSe]:SEMask:AVERage:COUNt <integer>
[:SENSe]:SEMask:AVERage:COUNt?
```

Specifies average number in SEM measurement.

Example: `:SEMask:AVERage:COUNt 20`

Key access: **Meas > SEM > Average Number**

SEM Average State

```
[:SENSe]:SEMask:AVERage[:STATe] OFF|ON|0|1
[:SENSe]:SEMask:AVERage[:STATe]?
```

Use this command to turn on/off the average in SEM measurement.

Example: `:SEMask:AVERage 1`

Key access: **Meas > SEM > Average Number**

SEM Channel Integrate Bandwidth

```
[:SENSe]:SEMask:BANDwidth|BWIDth:INTegration
<freq>
[:SENSe]:SEMask:BANDwidth|BWIDth:INTegration?
```

Use this command to specify reference channel integrate bandwidth in SEM measurement.

Example: `:SEMask:BANDwidth:INTegration 2 MHz`

Key access: **Meas > SEM > Ref Channel > Chan Integ BW**

SEM Resolution Bandwidth

```
[:SENSe]:SEMask:BANDwidth|BWIDth <freq>
[:SENSe]:SEMask:BANDwidth|BWIDth?
```

Use this command to specify resolution bandwidth of reference channel in SEM measurement.

```
[:SENSe]:SEMask:BANDwidth|BWIDth:AUTO OFF|ON|0|1
[:SENSe]:SEMask:BANDwidth|BWIDth:AUTO?
```

Automatically selects the resolution bandwidth to balance the measurement accuracy and test speed.

Example: `:SENSe:SEMask:BANDwidth 30 Hz`

Key access: **Meas > SEM > Ref Channel > Res BW**

SEM Total Power Reference

```
[:SENSe]:SEMask:CARRier[:POWer] <value>
[:SENSe]:SEMask:CARRier[:POWer]?
```

Use this command to specify the total power reference in SEM measurement. The unit of the amplitude is dBm.

```
[:SENSe]:SEMask:CARRier:AUTO[:STATe] OFF|ON|0|1
[:SENSe]:SEMask:CARRier:AUTO[:STATe]?
```

Automatically select the total power reference.

Example: `:SEMask:CARRier -40`

Key access: **Meas > SEM > Ref Channel > Total Pwr Ref**

SEM Channel Span

```
[:SENSe]:SEMask:FREQuency:SPAN <freq>
[:SENSe]:SEMask:FREQuency:SPAN?
```

Use this command to specify reference channel span in SEM measurement.

Example: `:SEMask:FREQuency:SPAN 20MHz`

Key access: **Meas > SEM > Ref Channel > Chan Span**

SEM Center Frequency

```
[:SENSe]:SEMask:FREQuency:CENTer <freq>
[:SENSe]:SEMask:FREQuency:CENTer?
```

Use this command to set center frequency in SEM measurement.

Example: `:SEMask:FREQuency:CENTer 1 GHz`

Key access: **Freq > Center Freq**

SEM Marker State

```
[:SENSe]:SEMask:MARKer[1]|2|3|4|5|6[:STATe]
OFF|ON|0|1
[:SENSe]:SEMask:MARKer[1]|2|3|4|5|6[:STATe]?
```

Use this command to turn on/off the markers in SEM measurement.

Example: `:SENSe:SEMask:MARKer2 ON`

Key access: **Marker**

SEM Marker X

```
[:SENSe]:SEMask:MARKer[1]|2|3|4|5|6:X <freq>
[:SENSe]:SEMask:MARKer[1]|2|3|4|5|6:X?
```

Use this command to specify the marker X axis value, or query the value of marker X axis.

Example: `:SENSe:SEMask:MARKer1:X 1.3 GHZ`

Key access: **Marker**

Query SEM Marker Y Value

```
[:SENSe]:SEMask:MARKer[1]|2|3|4|5|6:Y?
```

Use this command to query the marker readout of Y axis in SEM measurement.

Example: `:SEMask:MARKer1:Y?`

SEM Marker All Off

```
[:SENSe]:SEMask:MARKer:AOFF
```

Use this command to turn off all the marker in SEM measurement.

Example: `:SEMask:MARKer:AOFF`

Key access: **Marker > Off**

SEM Offset Resolution BW

```
[:SENSe]:SEMask:OFFSet:LIST:BAND-
width|BWIDth[:RESolution] <real_number>,
<real_number>, <real_number>,
<real_number>,<real_number>
[:SENSe]:SEMask:OFFSet:LIST:BAND-
width|BWIDth[:RESolution]?
```

Use this command to specify the resolution bandwidth of reference channels in SEM measurement.

[:SENSe]:SEMask:OFFSet:LIST:BANDwidth|BWIDth[:RESolution]:AUTO OFF|ON|0|1,...

[:SENSe]:SEMask:OFFSet:LIST:BANDwidth|BWIDth[:RESolution]:AUTO?

Use this command to turn on/off automatically setting of resolution bandwidth of reference channel in SEM measurement.

Example: `SENSe:SEMask:OFFSet:LIST:BANDwidth 1E6, 1E6, 1E6, 1E6,1E6`

Key access: **Meas > Offset/Limit > Res BW**

SEM Offset Measurement Bandwidth

```
[:SENSe]:SEMask:OFFSet:LIST:BAND-
width|BWIDth:MEASure <Integer>, <Integer>,
<Integer>, <Integer>,<Integer>
[:SENSe]:SEMask:OFFSet:LIST:BAND-
width|BWIDth:MEASure?
```

Use this command to set the measurement bandwidth of offset channel in SEM measurement.

Example: `:SENSe:SEMask:OFFSet:LIST:BANDwidth:MEASure 1,1,1,1,1`

Key access: **Meas > SEM > Offset/Limit > More > More > Meas BW**

SEM Offset Start Freq

```
[:SENSe]:SEMask:OFFSet:LIST:FREQuency:STARt
<freq>,<freq>,<freq>,<freq>,<freq>
[:SENSe]:SEMask:OFFSet:LIST:FREQuency:STARt?
```

Use this command to specify start frequency of offset channel in SEM measurement.

Example: `:SENSe:SEMask:OFFSet:LIST:FREQuency:STARt 2E6, 5E6, 10E6, 1E6, 2E6`

Key access: **Meas > SEM > Offset/Limit** > Start Freq

SEM Offset Stop Freq

```
[:SENSe]:SEMask:OFFSet:LIST:FREQuency:STOP
<freq>,<freq>,<freq>,<freq>,<freq>
[:SENSe]:SEMask:OFFSet:LIST:FREQuency:STOP?
```

Use this command to specify start frequency of offset channel in SEM measurement.

Example: `:SENSe:SEMask:OFFSet:LIST:FREQuency:STOP 3E6, 5E6, 8E6,15E6, 20E6`

Key access: **Meas > SEM > Offset/Limit > Stop Freq**

SEM Offset Sweep Time

```
[:SENSe]:SEMask:OFFSet:LIST:SWEeptime
<time>,<time>,<time>,<time>,<time>
[:SENSe]:SEMask:OFFSet:LIST:SWEeptime?
```

Use this command to specify the related sweep time of offset channels in SEM measurement. The unit for the time in this command is second.

```
[:SENSe]:SEMask:OFFSet:LIST:SWEeptime:AUTO
OFF|ON|0|1,...
[:SENSe]:SEMask:OFFSet:LIST:SWEeptime:AUTO?
```

Use this command to turn on/off automatically setting of sweep time of reference channels in SEM measurement.

Example: `:SENSe:SEMask:OFFSet:LIST:SWEeptime 10,10,10,10,10`

Key access: **Meas > SEM > Offset/Limit > Sweep Time**

SEM Offset Abs StartAmpt

```
[:SENSe]:SEMask:OFFSet:LIST:STARt:ABSolute
<ampl>,<ampl>,<ampl>,<ampl>,<ampl>
[:SENSe]:SEMask:OFFSet:LIST:STARt:ABSolute?
```

Use this command to specify the start frequency amplitude of offset channel absolute limit mask in SEM measurement. Before use this command, the fail mask type should be set to “Absolute” or “Abs AND Rel” or “Abs OR Rel”.

Example: `SENSe:SEMask:OFFSet:LIST:STARt:ABSolute -14, -14,-15,-20,-20`

Key access: **Meas > SEM > Offset/Limit > More 1 of 3 > Abs StratAmpt**

SEM Offset Rel StartAmpt

```
[:SENSe]:SEMask:OFFSet:LIST:STARt:RCARrier
<ampl>,<ampl>,<ampl>,<ampl>,<ampl>
[:SENSe]:SEMask:OFFSet:LIST:STARt:RCARrier?
```

Use this command to specify the start frequency amplitude of offset channel relative limit mask in SEM measurement. Before use this command, the fail mask type should be set to “Relative” or “Abs AND Rel” or “Abs OR Rel”.

Example: `SENSe:SEMask:OFFSet:LIST:STARt:RCARrier -14, -14, -15,-20, -20`

Key access: **Meas > SEM > Offset/Limit > More 1 of 3 > Rel StratAmpt**

SEM Offset Abs Stop Ampt

```
[:SENSe]:SEMask:OFFSet:LIST:STOP:ABSolute
<ampl>,<ampl>,<ampl>,<ampl>,<ampl>
[:SENSe]:SEMask:OFFSet:LIST:STOP:ABSolute?
```

Use this command to specify the stop frequency amplitude of offset channel absolute limit mask in SEM measurement. Before use this command, the fail mask type should be set to “Absolute” or “Abs AND Rel” or “Abs OR Rel”.

Example: `:SENSe:SEMask:OFFSet:LIST:STOP:ABSolute -10, -20, -20, -30, -30`

Key access: **Meas > SEM > Offset/Limit > More 1 of 3 > Abs Stop Ampt**

SEM Offset Rel StopAmpt

```
[:SENSe]:SEMask:OFFSet:LIST:STOP:RCARrier
<ampl>,<ampl>,<ampl>,<ampl>,<ampl>
[:SENSe]:SEMask:OFFSet:LIST:STOP:RCARrier?
```

This command defines the stop frequency amplitude of offset channel relative limit mask in SEM measurement. The fail mask type should be set to “Relative” or “Abs AND Rel” or “Abs OR Rel”.

Example: `SENSe:SEMask:OFFSet:LIST:STOP:RCARrier -10, -20, -20, -30, -30`

Key access: **Meas > SEM > Offset/Limit > More 1 of 3 > Rel Stop Ampt**

SEM Offset Fail Mask

```
[:SENSe]:SEMask:OFFSet:LIST:TEST ABSolute|RELa-
tive|AND|OR,ABSolute|RELative|AND|OR,ABSo-
lute|RELative|AND|OR,ABSolute|RELative|AND|OR,AB
Solute|RELative|AND|OR
[:SENSe]:SEMask:OFFSet:LIST:TEST?
```

Use this command to choose the mask type in SEM measurement.

Range: ABSolute, RELative, AND, OR

Example:
```
:SENSe:SEMask:OFFSet:LIST:TEST
ABSolute,ABSolute, RELative, AND, OR
```

Key access: **Meas > SEM > Offset/Limit > More 1 of 3 > More 2 of 3 > Fail Mask**

SEM Meas Type

```
[:SENSe]:SEMask:TYPE TPRef|PSDRef
[:SENSe]:SEMask:TYPE?
```

Use this command to switch the measurement type in SEM measurement.

Example:
```
SENSe:SEMask:TYPE PSDRef
```

Key access: **Meas > SEM > Meas Type**

SEM Lower Offset Fail Flag

```
[:SENSe]:SEMask:OFFSet:LIST:LOWer:FAIL?
```

Use this command to query fail flag of each lower offset channel in SEM measurement.

Example:
```
:SEMask:OFFSet:LIST:LOWer:FAIL?
```

SEM Lower Offset Power

```
[:SENSe]:SEMask:OFFSet:LIST:LOWer:POWer?
```

Use this command to query the measured power of each lower offset channel in SEM measurement.

Example:
```
:SEMask:OFFSet:LIST:LOWer:POWer?
```

Query SEM Lower Offset Peak Frequency

```
[:SENSe]:SEMask:OFFSet:LIST:LOWer:PEAK:FRE-
Quency?
```

Use this command to query the peak power frequency of each lower offset channel in SEM measurement.

Example: `:SEMask:OFFSet:LIST:LOWer:PEAK:FREQuency?`

Query SEM Upper Offset Fail Flag

```
[:SENSe]:SEMask:OFFSet:LIST:Upper:FAIL?
```

Use this command to query fail flag of each upper offset channel in SEM measurement.

Example: `:SEMask:OFFSet:LIST:UPPer:FAIL?`

Query SEM Upper Offset Power

```
[:SENSe]:SEMask:OFFSet:LIST:UPPer:POWer?
```

Use this command to query the measured power of each upper offset channel in SEM measurement.

Example: `:SEMask:OFFSet:LIST:UPPer:POWer?`

Query SEM Upper Offset Peak Frequency

```
[:SENSe]:SEMask:OFFSet:LIST:Upper:PEAK:FRE-
Quency?
```

Use this command to query the peak power frequency of each upper offset channel in SEM measurement.

Example: `:SEMask:OFFSet:LIST:LOWer:PEAK:FREQuency?`

SEM Fail Setting Log

```
[:SENSe]:SEMask:FAIL:LOG OFF|ON|0|1
[:SENSe]:SEMask:FAIL:LOG?
```

Use this command to switch the Fail log status of SEM measurement.

Example: `:SEMask:FAIL:LOG ON`

Key access: **Meas > SEM > Fail Setting > Log**

SEM Fail Setting Hold

```
[:SENSe]:SEMask:FAIL:LOG OFF|ON|0|1
[:SENSe]:SEMask:FAIL:LOG?
```

Use this command to switch the Fail log status of SEM measurement.

Example: `:SEMask:FAIL:LOG ON`

Key access: **Meas > SEM > Fail Setting > Log**

SEM Fail Setting Hold

```
[:SENSe]:SEMask:FAIL:HOLD OFF|ON|0|1
[:SENSe]:SEMask:FAIL:HOLD?
```

Use this command to switch the Fail hold status of SEM measurement.

Example: `:SEMask:FAIL:HOLD ON`

Key access: **Meas > SEM > Fail Setting > Hold**

SEM Fail Setting Beep

```
[:SENSe]:SEMask:FAIL:BEEP OFF|ON|0|1
[:SENSe]:SEMask:FAIL:BEEP?
```

Use this command to switch the Fail log status of SEM measurement.

Example: `:SEMask:FAIL:BEEP ON`

Key access: **Meas > SEM > Fail Setting > Beep**

Spectrum Monitor Option Subsystem

The spectrum monitor subsystem provides you the SCPI commands reference for the spectrum monitor with spectrogram record and playback function. Use the command `:INSTrument SA`, and `:INSTrument:MEASure SPECtrogram` to access this subsystem.

Option MNT is required to enable this function.

Spectrogram State

```
[:SENSe]:SPECtrogram:STATe RUN|PAUSe
[:SENSe]:SPECtrogram:STATe?
```

Use this command to run or pause spectrogram measurement.

Example: `:SPECtrogram:STATe PAUSe`

Key access: **Meas > Spectrogram Run/Pause**

Spectrogram Restart Measurement

```
[:SENSe]:SPECtrogram:RESTart
```

Use this command to restart spectrogram measurement.

Example: `:SPECtrogram:RESTart`

Key access: **Meas > Restart Meas**

Spec Update Interval

```
[:SENSe]:SPECtrogram:UINTerval <value>
[:SENSe]:SPECtrogram:UINTerval?
```

This command defines the update interval of spectrogram measurement.

Example: `:SPECtrogram:UINTerval 1 s`

```
[:SENSe]:SPECtrogram:UINTerval:STATe OFF|ON|0|1
[:SENSe]:SPECtrogram:UINTerval:STATe?
```

Use this command to turn on/off update interval of current test.

Example: `:SPECtrogram:UINTerval:STATe ON`

Key access: **Meas > Update Int.**

Spectrogram Palette

```
[:SENSe]:SPECtrogram:PALette FCOLor|GRAY
[:SENSe]:SPECtrogram:PALette?
```

This command switches the display palette between Color and Gray.

Example: `:SPECtrogram:PALette GRAY`

Key access: **Meas > Palette > Color /Gray**

Spectrogram Display Mode

```
[:SENSe]:SPECtrogram:DMODe SPCTrg|TRACe|BOTH
[:SENSe]:SPECtrogram:DMODe?
```

Use this command to switch the spectrogram display mode between Spectrogram, Trace and both display.

Range: SPEC, TRACe, BOTH

Example: `:SPECtrogram:DMODe BOTH`

Key access: **Meas > Disp Mode**

Spectrogram Marker State

```
[:SENSe]:SPECtrogram:MARKer[1]|2:STATe
OFF|ON|0|1
[:SENSe]:SPECtrogram:MARKer[1]|2:STATe?
```

Use this command to turn on/off the marker in current test.

Example: `:SPECtrogram:MARKer2:STATe ON`

Key access: **Marker > State**

Spectrogram Display Frequency

```
[:SENSe]:SPECtrogram:MARKer[1]|2:FREQuency
<value>
[:SENSe]:SPECtrogram:MARKer[1]|2:FREQuency?
```

Use this command to set the frequency of marker in spectrogram measurement.

Example: `:SPECtrogram:MARKer2:FREQuency 1GHz`

Key access: **Marker > Frequency**

Spectrogram Marker Time

```
[:SENSe]:SPECtrogram:MARKer[1]|2:TIME <value>
[:SENSe]:SPECtrogram:MARKer[1]|2:TIME?
```

Use this command to locate the marker to specific time point in spectrogram measurement.

Example: `:SPECtrogram:MARKer1:TIME 1 ks`

Key access: **Marker > Time**

Spectrogram Marker Peak Search

```
[:SENSe]:SPECtrogram:MARKer[1]|2:PEAK
```

Use this command to perform Marker peak search once.

Example: `:SPECtrogram:MARKer1:PEAK`

Key access: **Marker > Peak Search**

Spectrogram Marker Previous Failure Search

```
[:SENSe]:SPECtrogram:MARKer[1]|2:FAIL:PRE
```

Use this command to locate Marker to the previous fail trace (frame) in spectrogram measurement result.

Example: `:SPECtrogram:MARKer1:FAIL:PRE`

Key access: **Marker > Failed Frame Search**

Spectrogram Marker Next Failure Search

```
[:SENSe]:SPECtrogram:MARKer[1]|2:PEAK
```

Use this command to locate Marker to the next fail trace (frame) in spectrogram measurement result.

Example: `:SPECtrogram:MARKer1:FAIL:NEXT`

Key access: **Marker > Failed Frame Search**

Spectrogram Marker Band Power State

```
[:SENSe]:SPECtrogram:MARKer:BPOWer:STATe
OFF|ON|0|1
[:SENSe]:SPECtrogram:MARKer:BPOWer:STATe?
```

Use this command to turn on/off the band power test of the two Markers in spectrogram measurement.

Example: `:SPECtrogram:MARKer:BPOWer:STATe ON`

Key access: **Marker > More 1 of 2 > Band Power**

Spectrogram Marker Band Power Value

```
[:SENSe]:SPECtrogram:MARKer:BPOWer?
```

Use this command to query the Marker band power measurement result.

Example: `:SPECtrogram:MARKer:BPOWer?`

Spectrogram Marker Band Width Value

```
[:SENSe]:SPECtrogram:MARKer:BWIDth?
```

Use this command to query the width of band power tested.

Example: `:SPECtrogram:MARKer:BWIDth?`

Spectrogram Marker Audio Alert State

```
[:SENSe]:SPECtrogram:MARKer:ALERt:STATe
OFF|ON|0|1
[:SENSe]:SPECtrogram:MARKer:ALERt:STATe?
```

Use this command to turn on or off the Marker audio alert state. If the Marker audio alert state is set to on, N9322C will alert when there is signal beyond the audio alert limit value.

Example: `:SPECtrogram:MARKer:ALERt:STATe ON`

Key access: **Meas > Spectrum Monitor, then Marker > More 1 of 2 > Audio Alert**

Spectrogram Marker Audio Alert Limit

```
[:SENSe]:SPECtrogram:MARKer:ALERt:LIMit <value>
[:SENSe]:SPECtrogram:MARKer:ALERt:LIMit?
```

Use this command to set the Marker audio alert limit line value.

Example: `:SPECtrogram:MARKer:ALERt:LIMit -50 dBm`

Key access: **Marker > More 1 of 2 > Audio Alert**

Spectrogram Start Save

```
[:SENSe]:SPECtrogram:SAVE:STARt
[:SENSe]:SPECtrogram:SAVE[:STATe]?
```

Use this command to start to save the spectrogram measurement result.

Example: `:SPECtrogram:SAVE:STARt`

Key access: **Meas > File Logging > Start Save**

Spectrogram Stop Save

```
[:SENSe]:SPECtrogram:SAVE:STOP
[:SENSe]:SPECtrogram:SAVE[:STATe]?
```

Use this command to stop to save the spectrogram measurement result.

Example: `:SPECtrogram:SAVE:STOP`

Key access: **Meas > File Logging > Stop Save**

Spectrogram Time Save State

```
[:SENSe]:SPECtrogram:SAVE:TIMed[:STATe]
OFF|ON|0|1
[:SENSe]:SPECtrogram:SAVE:TIMed[:STATe]?
```

Use this command to turn on/off timed save state of spectrogram measurement.

Example: `:SPECtrogram:SAVE:TIMed ON`

Key access: **Meas > File Logging > Time Setting > Timed Save**

Spectrogram Save Start Date

```
[:SENSe]:SPECtrogram:SAVE:TIMed:STARt:DATE
<"str">
[:SENSe]:SPECtrogram:SAVE:TIMed:STARt:DATE?
```

Use this command to set the start date of timed save operation.

Example: `:SPECtrogram:SAVE:TIMed:STARt:DATE "20121016"`

Key access: **Meas > File Logging > Time Setting > Start Date**

Spectrogram Save Start Time

```
[:SENSe]:SPECtrogram:SAVE:TIMed:STARt:TIME
<"str">
[:SENSe]:SPECtrogram:SAVE:TIMed:STARt:TIME?
```

Use this command to set the start time of timed save operation.

Example: `:SPECtrogram:SAVE:TIMed:STARt:TIME "012403"`

Key access: **Meas > File Logging > Time Setting > Start Time**

Spectrogram Save Stop Date

```
[:SENSe]:SPECtrogram:SAVE:TIMed:STOp:DATE
<"str">
[:SENSe]:SPECtrogram:SAVE:TIMed:STOp:DATE?
```

Use this command to set the stop date of timed save operation.

Example: `:SPECtrogram:SAVE:TIMed:STOp:DATE "20121016"`

Key access: **Meas > File Logging > Time Setting > Stop Date**

Spectrogram Save Stop Time

```
[:SENSe]:SPECtrogram:SAVE:TIMed:STOp:TIME
<"str">
[:SENSe]:SPECtrogram:SAVE:TIMed:STOp:TIME?
```

Use this command to set the stop time of timed save operation.

Example: `:SPECtrogram:SAVE:TIMed:STOp:TIME "012403"`

Key access: **Meas > Spectrum Monitor > File Logging > Time Setting > Stop Time**

Spectrogram Save File Type

```
[:SENSe]:SPECtrogram:SAVE:FTYPe TRC|CSV
[:SENSe]:SPECtrogram:SAVE:FTYPe?
```

Use this command to switch the spectrogram test result type between TRC and CSV.

Example: `:SPECtrogram:SAVE:FTYPe TRC`

Key access: **Meas > File Logging > File Type**

Spectrogram Save File Size

```
[:SENSe]:SPECtrogram:SAVE:FSIZe <int>
[:SENSe]:SPECtrogram:SAVE:FSIZe?
```

Use this command to set the maximum number of trace (frame) saved in one spectrogram test result file.

Range: 1 to 1500 frames

Example: `:SPECtrogram:SAVE:FSIZe 1400`

Key access: **Meas > Spectrum Monitor > File Logging > File Size**

Spectrogram Marker Y Value

```
[:SENSe]:SPECtrogram:MARKer[1]|2:Y?
```

Use this command to query the Marker Y axis measurement result in spectrogram measurement.

Example: `:SPECtrogram:MARKer:Y?`

Reflection Measurement Option Subsystem

The Reflection Measurement Option Subsystem provides you the SCPI command reference for the reflection measurement function. Use the command `:INSTrument CAT` to access this subsystem.

Option RM7 and TG7 are required to enable the reflection measurement function.

Reflection Measurement Type

`[:SENSe]:CAT REFLection|OPCLoss|DTF`

`[:SENSe]:CAT?`

Toggles the reflection measurement mode between Reflection Measurement, One Port Insertion LossMeasurement and Distance to Fault Measurement.

Range: REFLection, OPCLoss, DTF

Example: `:CAT OPCL`

Key access: **Meas > Meas > Reflection Measurement / One Port Insertion Loss/ Distance To Fault**

Reflection Measurement Calibration Type

`[:SENSe]:CAT:CALibrate:TYPE FULL|SELected`

`[:SENSe]:CAT:CALibrate:TYPE?`

This command toggles calibration type between full frequency range calibration and selected frequency range calibration.

Range: FULL, SELected

Example: `:CAT:CALibrate:TYPE FULL`

Key access: **Freq > Cal Type**

Reflection Measurement IIM State

```
[:SENSe]:CAT:IIM[:STATe] OFF|ON|0|1
[:SENSe]:CAT:IIM[:STATe]?
```

Use this command to turn on/off the IIM state for reflection measurement.

Example: `:CAT:IIM ON`

Key access: **Meas > IIM**

Reflection Measurement Display Type

```
[:SENSe]:CAT:DISPlay:TYPE RL|VSWR|LIN
[:SENSe]:CAT:DISPlay:TYPE?
```

Toggles the reflection measurement display mode between RL, VSWR and LIN. The RL stands for Return Loss, and LIN stands for Reflection Coefficient. It's only available for Reflection Measurement mode, and Distance to Fault measurement mode.

Range: RL, VSWR, LIN

Example: `:CAT:DISPlay:TYPE VSWR`

Key access: **Meas > Disp Mode**

Set Reflection Measurement Cable Type

```
[:SENSe]:CAT:CABLe:TYPE S1000| S2000| S2500|CUST
[:SENSe]:CAT:CABLe:TYPE
```

Toggles the cable type between S1000, S2000, S2500 and CUST. CUST stands for custom cable type. It's only available for Distance To Fault measurement.

Range: S1000, S2000, S2500, CUST

Example: `:CAT:CABLe:TYPE S1000`

Key access: **Meas > More 1 of 2 > Cable Specification > Cable Type**

Select A Cable in Reflection Measurement

```
[:SENSe]:CAT:CABLe:LOAD <string>
```

Use this command to select a cable from built-in cable list.

Example: `:CAT:CABLe:LOAD "5088"`

Key access: **Meas > More 1 of 2 > Cable Specification > Select Cable**

Reflection Measurement Cable Attenuation

```
[:SENSe]:CAT:CABLe:ATTenuation <value>
[:SENSe]:CAT:CABLe:ATTenuation?
```

This command defines the attenuation value for a custom cable.

Range: 0 to 5 dB/m, or 0 to 1.524 dB/ft

Example: `:CAT:CABLe:ATTenuation 1.2`

Key access: **Meas > More 1 of 2 > Cable Specification > Cable Type > Custom, then Cable Atten**

Reflection Measurement Cable Velocity Factor

```
[:SENSe]:CAT:CABLe:VELFactor <value>
[:SENSe]:CAT:CABLe:VELFactor?
```

This command defines the velocity factor for a custom cable.

Range: 1 to 100

Example: `:CAT:CABLe:VELFactor 89.2`

Key access: **Meas > More 1 of 2 > Cable Specification > Cable Type > Custom, then Vel Factor**

Reflection Measurement Windows Select

```
[:SENSe]:CAT:WINDow
NONE|RECTangular|HAMMing|S3BLackman|S4BLackman
[:SENSe]:CAT:WINDow?
```

This command toggles the measurement window between Rectangular Window, Hamming Window, 3-Sample Blackman Window and 4-Sample Blackman Window.

Range: NONE, RECTangular, HAMMing, S3BLackman, S4BLackman

Example: `:CAT:WINDow RECT`

Key access: **Meas > More 1 of 2 > Window**

Reflection Measurement Start Frequency

```
[:SENSe]:CAT:FREQuency:STARt <freq>
[:SENSe]:CAT:FREQuency:STARt?
```

Use this command to define the start frequency of reflection measurement.

Range: 5 MHz to 7 GHz

Example: `:CAT:FREQuency:STARt 200 MHz`

Key access: **Freq > Start Frequency**

Reflection Measurement Stop Frequency

```
[:SENSe]:CAT:FREQuency:STOP <freq>
[:SENSe]:CAT:FREQuency:STOP?
```

Use this command to define the stop frequency of reflection measurement.

Range: 5 MHz to 7 GHz

Example: `:CAT:FREQuency:STOP 1 GHz`

Key access: **Freq > Stop Frequency**

Reflection Measurement Center Frequency

```
[:SENSe]:CAT:FREQuency:CENTer <freq>
[:SENSe]:CAT:FREQuency:CENTer?
```

Use this command to define the center frequency of reflection measurement.

Range: 5 MHz to 7 GHz

Example: `:CAT:FREQuency:CENTer 200 MHz`

Key access: **Freq > Center Freq**

Reflection Measurement Frequency Span

```
[:SENSe]:CAT:FREQuency:SPAN <freq>
[:SENSe]:CAT:FREQuency:SPAN?
```

This command defines the frequency span of current measurement.

Example: `;CAT:FREQuency:SPAN 200 MHz`

Key access: **Span > Span**

Reflection Measurement Distance Start

```
[:SENSe]:CAT:DISTance:STARt <distance>
[:SENSe]:CAT:DISTance:STARt?
```

This command defines the start distance point of DTF measurement

Range: 0 to Stop distance - 0.1

Example: `:CAT:DISTance:STARt 1.3`

Key access: **Meas > Start Distance**

Reflection Measurement Distance Stop

```
[:SENSe]:CAT:DISTance:STOP <distance>
[:SENSe]:CAT:DISTance:STOP?
```

This command defines the stop distance of DTF measurement

Range: 0 to stop distance – 0.1 meter

Example: `:CAT:DISTance:STOP 100.3`

Key access: **Meas > Stop Distance**

Reflection Measurement Distance Unit

`[:SENSe]:CAT:DISTance:UNIT METers|FEET`

`[:SENSe]:CAT:DISTance:UNIT?`

Use this command to toggle the distance unit between meter and feet.

Example: `:CAT:DISTance:UNIT FEET`

Key access: **Meas > Unit > Meter/Feet**

Reflection Measurement DTF Frequency Domain State

`[:SENSe]:CAT:DTF:FVIew OFF|ON|0|1`

`[:SENSe]:CAT:DTF:FVIew?`

Use this command to turn on/off frequency domain display in distance to fault measurement mode.

Example: `:CAT:DTF:FVIew ON`

Key access: **Meas > Freq Domain**

Reflection Measurement Amplitude Auto Scale

`CAT:SCALe:AUTO`

Use this command to perform amplitude auto scale once.

Example: `:CAT:SCALe:AUTO`

Key access: **Amptd > Auto Scale**

Reflection Measurement Amplitude Scale Reference Level

`[:SENSe]:CAT:SCALe:REFerence <value>`

`[:SENSe]:CAT:SCALe:REFerence?`

This command defines the amplitude scale reference level.

Range: 0 to 60

Example: `:CAT:SCALe:REFerence 12`

Key access: **Amptd > Ref Level**

Reflection Measurement Amplitude Scale Per Division

`[:SENSe]:CAT:SCALe:DIV <value>`

`[:SENSe]:CAT:SCALe:DIV?`

This command defines the amplitude display scale per division.

Range: 0.1 to 10

Example: `:CAT:SCALe:DIV 5`

Key access: **Meas > Start Distance**

Reflection Measurement Save Trace Data to Memory

`[:SENSe]:CAT:TRACe:SAVE`

Use this command to save the current trace data to memory.

Example: `:CAT:TRACe:SAVE`

Key access: **Trace > Data >Mem**

Reflection Measurement Trace Display Type

`[:SENSe]:CAT:TRACe:DISPlay DATA|MEM|DAM`

`[:SENSe]:CAT:TRACe:DISPlay?`

This command toggles the trace display type. Reflection measurement supports three display type, displaying current measurement result, displaying the trace data saved in memory, and display both the data saved in memory and current sweep result.

Range: DATA, MEM, DAM

Example: `:CAT:TRACe:DISPlay MEM`

Key access: **Trace> Data / Memory / Data&Memory**

Reflection Measurement Trace Math Type

```
[:SENSe]:CAT:TRACe:MATH OFF|ADD|SUB|DIV
[:SENSe]:CAT:TRACe:MATH?
```

This command toggles the trace math type of reflection measurement. RM7 option support three trace type, measurement trace data plus memory saved trace data, immurement trace data minus memory saved trace data, and memory saved data minus measurement trace data.

Example: `:CAT:TRACe:MATH ADD`

Key access: **Trace > Trace Math**

Reflection Measurement Trace Average Number

```
[:SENSe]:CAT:AVERage:COUNt <value>
[:SENSe]:CAT:AVERage:COUNt?
```

This command defines the trace average number of CAT test.

```
[:SENSe]:CAT:AVERage[:STATe] OFF|ON|0|1
[:SENSe]:CAT:AVERage[:STATe]?
```

Use this command to turn on/off the trace average operation.

Example: `:CAT:AVERage:COUNt 20`

Key access: **Trace > Average Number**

Reflection Measurement Trace Data Format

```
:FORMat[:TRACe][:DATA] ASCii|REAL
:FORMat[:TRACe][:DATA]?
```

This command toggles the trace data format. This command affects only return trace data format. ASC stands for ASCII string be separated by comma, and REAL stands for 4 bytes length float without comma.

Range: ASCii, REAL

Example: `:FORMat ASC`

Reflection Measurement Trace Data Query

```
:TRACe[:DATA]? DATA|MEM
```

Use this command to query the current trace data.

Example: `:TRACe? DATA`

Reflection Measurement Sweep Continuos State

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

Use this command to turn on/off continuos sweep state of current test.

Example: `:INITiate:CONTinuous 1`

Key access: **Shift > Sweep> Sweep > Single / Cont**

Reflection Measurement Single Sweep

```
:INITiate[:IMMediate]
```

Use this command to perform single sweep once.

Example: `:INITiate`

Key access: **Shift > Sweep> Sweep > Single Sweep**

Reflection Measurement Marker All Off

```
:CALCulate:MARKer:AOFF
```

This command turns off all the Marker displayed on screen.

Example: `:CALCulate:MARKer:AOFF`

Key access: **Marker > More 1 of 2 > All Off**

Reflection Measurement Marker Mode

```
:CALCulate:MARKer[1]|2|3|4|5|6:MODE
OFF|POSition|DELTa
:CALCulate:MARKer[1]|2|3|4|5|6:MODE?
```

This command toggles the marker mode in CAT test.

Example: `:CALCulate:MARKer1:MODE DELTa`

Key access: **Marker > Normal/Delta/Off**

Reflection Measurement Marker Continuous Peak State

```
:CALCulate:MARKer[1]|2|3|4|5|6:CPEak OFF|ON|0|1
:CALCulate:MARKer[1]|2|3|4|5|6:CPEak?
```

This command turns on/off marker continuous peak state in reflection measurement.

Example: `:CALCulate:MARKer1:CPEak ON`

Key access: **Marker > Cont Peak**

Reflection Measurement Marker Continuous Valley State

```
:CALCulate:MARKer[1]|2|3|4|5|6:CVALley
OFF|ON|0|1
:CALCulate:MARKer[1]|2|3|4|5|6:CVALley?
```

This command turns on/off marker continuous valley state in reflection measurement.

Example: `:CALCulate:MARKer1:CVALley ON`

Key access: **Marker > Cont Valley**

Reflection Measurement Set Marker X axis Value

```
:CALCulate:MARKer[1]|2|3|4|5|6:X <value>
:CALCulate:MARKer[1]|2|3|4|5|6:X?
```

This command sets current Marker X axis value.

Range: current frequency range or distance range

Example: `:CALCulate:MARKer1:X 3500000000`

Key access: **Marker > Normal, then input the value with the numeric keys and soft keys**

Query Marker Y axis Value in Reflection Measurement

```
:CALCulate:MARKer[1]|2|3|4|5|6:Y?
```

Use this command to query current Marker Y axis value readout.

Example: `:CALCulate:MARKer1:Y?`

Query Reference Marker X axis Value in Reflection Measurement

`:CALCulate:MARKer[1]|2|3|4|5|6:RX?`

This command query current reference marker X axis value.

Example: `:CALCulate:MARKer1:RX?`

Query Reference Marker Y axis Value in Reflection Measurement

`:CALCulate:MARKer[1]|2|3|4|5|6:RY?`

This command query current reference marker Y axis value.

Example: `:CALCulate:MARKer1:RY?`

Marker Peak Search in Reflection Measurement

`:CALCulate:MARKer[1]|2|3|4|5|6:PEAK`

Use this command to perform marker peak search once.

Example: `:CALCulate:MARKer1:PEAK`

Key access: **Shift > Peak > Peak**

Marker Valley Search in Reflection Measurement

`:CALCulate:MARKer[1]|2|3|4|5|6:VALLey`

Use this command to perform marker valley search once. The maker valley search can find the point with the lowest value in the current trace.

Example: `:CALCulate:MARKer1:VALL`

Key access: **Shift > Peak > Valley**

Reflection Measurement Limit Beep State

`:CALCulate:MARKer[1]|2|3|4|5|6:CPEak OFF|ON|0|1`

`:CALCulate:MARKer[1]|2|3|4|5|6:CPEak?`

This command turns on/off the limit beep warning state in reflection measurement.

Example: `:CALCulate:MARKer1:CPEak ON`

Key access: **Shift > Limit > Beep**

Reflection Measurement Limit Type

```
:CALCulate:LLINe[1]|2:TYPE UPPer|LOWer
:CALCulate:LLINe[1]|2:TYPE?
```

This command toggles the limit type between upper limit and lower limit.

Example: `:CALCulate:LLINe:TYPE UPPer`

Key access: **Shift > Limit > Limit1 > Type**

Reflection Measurement Limit Line State

```
:CALCulate:LLINe[1]|2:STATe OFF|ON|0|1
:CALCulate:LLINe[1]|2:STATe?
```

This command turns on/off the limit line in CAT test.

Example: `:CALCulate:LLINe:STATe ON`

Key access: **Shift > Limit > Limit1 > Limit Line On/Off**

Reflection Measurement Limit Line Value

```
:CALCulate:LLINe[1]|2:Y <value>
:CALCulate:LLINe[1]|2:Y?
```

This command defines the limit line value in reflection measurement.

Example: `:CALCulate:LLINe:Y 15`

Key access: **Shift > Limit > Limit1 > Limit Line On/Off**

Reflection Measurement Limit State

```
:CALCulate:LLINe[1]|2:DISPlay OFF|ON|0|1
:CALCulate:LLINe[1]|2:DISPlay?
```

This command turns on/off the limit in reflection measurement. This limit is different from the limit line describe above, it is combined by several limit line with different amplitudes and different frequency points.

Example: `:CALCulate:LLINe1:DISPlay 1`

Key access: **Shift > Limit > Limit1 > Limit On/Off**

Get Limits Result in Reflection Measurement

```
:CALCulate:LLINe[1]|2:FAIL?
```

This command query the result of limit line or limit in reflection measurement.

Example:
```
:CALCulate:LLINe1:FAIL?
```

Define Limit Value in Reflection Measurement

```
:CALCulate:LLINe[1]|2:DATA
<x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<co
nnected>}
:CALCulate:LLINe[1]|2:DATA?
```

Use this command to define the limit value in reflection measurement function.

Example:
```
:CALC:LLIN1:DATA
1000000000,-20,0,200000000,-30,1
```

Key access: **Shift > Limit > Limit1 > Edit Limits**

Delete Limit Data in Reflection Measurement

```
:CALCulate:LLINe[1]|2:DELete
```

This command delete the current defined limit value.

Example:
```
:CALCulate:LLINe1:DELete
```

Get Limits Result in Reflection Measurement

```
:CALCulate:LLINe[1]|2:FAIL?
```

This command query the result of limit line or limit in reflection measurement mode.

Example:
```
:CALCulate:LLINe1:FAIL?
```

Define Limit Value in Reflection Measurement

```
:CALCulate:LLINe[1]|2:DATA
<x-axis>,<ampl>,<connected>{,<x-axis>,<ampl>,<co
nnected>}
:CALCulate:LLINe[1]|2:DATA?
```

This command defines the limit value in reflection measurement.

Example:
```
:CALC:LLIN1:DATA
1000000000,-20,0,200000000,-30,1
```

Key access: **Shift > Limit > Limit1 > Edit Limits**

Channel Scanner Option Subsystem

The Channel Scanner Option Subsystem provides you the SCPI command reference for the channel scanner function. Use the command `:INSTrument SA`, and command `:INSTrument:MEASure CHScanner` to access this subsystem.

Option SCN is required to enable the channel scanner function.

Channel Scanner SWEep Subsection

Sweep Mode

`:CHSCanner:SWEep:MODE LIST|TOP|BOTtom`

`:CHSCanner:SWEep:MODE?`

Toggles the channel scanner sweep mode between List mode, Top N mode and Bottom N mode.

Range: LIST, TOP, BOTtom

Example: `:CHSCanner:SWEep:MODE LIST`

Key access: **Meas > Channel Scanner > Scan Mode**

Top N Number

`:CHSCanner:SWEep:TOP <value>`

`:CHSCanner:SWEep:TOP?`

This command specify the scanning channel number of Top N mode. Need to switch to Top N sweep mode before using this command.

Range: 1 to 20

Example: `:CHSCanner:SWEep:TOP 10`

Key access: **Meas > Channel Scanner > Scan Mode > Top N**

Bottom N Number

```
:CHSCanner:SWEep:BOTTom <value>
:CHSCanner:SWEep:BOTTom?
```

This command specify the scanner channel number of Bottom N mode. Need to switch to Bottom N sweep mode before using this command.

Range: 1 to 20

Example: `:CHSCanner:SWEep:BOTTom 10`

Key access: **Meas > Channel Scanner > Scan Mode > Bottom N**

Max Hold State

```
:CHSCanner:SWEep:MAXHold[:STATe] OFF|ON|0|1
:CHSCanner:SWEep:MAXHold[:STATe]?
```

Use this command to turn on/off the trace max hold state.

Example: `:CHSCanner:SWEep:MAXHold ON`

Key access: **Trace > Max Hold**

Max Hold Dwell Time

```
:CHSCanner:SWEep:MAXHold:DWELltime <value>
:CHSCanner:SWEep:MAXHold:DWELltime?
```

Use this command to define the max hold dwell time of channel scanner measurement mode.

Range: 100 ms to 3600 s

Example: `:CHSCanner:SWEep:MAXHold:DWELltime 10s`

Key access: **Trace > Max Hold**

Mini Hold State

```
:CHSCanner:SWEep:MINHold[:STATe] OFF|ON|0|1
:CHSCanner:SWEep:MINHold [:STATe]?
```

Use this command to turn on/off min hold state.

Example: `:CHSCanner:SWEep:MINHold ON`

Key access: **Trace > Min Hold**

Min Hold Dwell Time

```
:CHSCanner:SWEep:MINHold:DWELltime <value>
:CHSCanner:SWEep:MINHold:DWELltime?
```

Use this command to define the min hold dwell time of channel scanner measurement mode.

Example: `:CHSCanner:SWEep:MINHold:DWELltime 10s`

Key access: **Trace > Min Hold**

Average State

```
:CHSCanner:SWEep:AVERage[:STATe] OFF|ON|0|1
:CHSCanner:SWEep:AVERage[:STATe]?
```

Use this command to turn on/off average state in channel scanner mode. It's only available for List sweep mode.

Example: `:CHSCanner:SWEep:AVERage ON`

Key access: **Trace > Average**

Average Count Number

```
:CHSCanner:SWEep:AVERage:COUNt <value>
:CHSCanner:SWEep:AVERage:COUNt?
```

Use this command to define the average count number.

Range: 1 to 1024

Example: `:CHSCanner:SWEep:AVERage:COUNt 1024`

Key access: **Trace > Average**

Average Mode

```
:CHSCanner:SWEep:AVERage:MODE EXP|REPeat
:CHSCanner:SWEep:AVERage:MODE?
```

Toggle the average mode between EXP mode and Repeat mode

Example: `:CHSCanner:SWEep:AVERage:MODE EXP`

Key access: **Trace > Average Mode**

Measurement Interval State

```
:CHSCanner:SWEep:INTerval[:STATe] OFF|ON|0|1
:CHSCanner:SWEep:INTerval [:STATe]?
```

Use this command to turn on/off measurement interval state.

Example: `:CHSCanner:SWEep:INTerval ON`

Key access: **Meas > Setup > Meas Interval**

Measurement Interval Time

```
:CHSCanner:SWEep:INTerval:TIME <value>
:CHSCanner:SWEep:INTerval:TIME?
```

Use this command to define the measurement interval time.

Range: 100 ms to 3600 s

Example: `:CHSCanner:SWEep:INTerval:TIME 10s`

Key access: **Meas > Setup > Interval Type**

Top N/Bottom N Measurement Rule

```
:CHSCanner:SWEep:TOPBottom:RULE:MEASure
SPEed|ACCUracy
:CHSCanner:SWEep:TOPBottom:RULE:MEASure?
```

Use this command to toggle the top/bottom N measurement rule between speed and accuracy.

Example: `:CHSCanner:SWEep:TOPBottom:RULE:MEASure ACCUracy`

Key access: **Meas > Setup > Measurement Rule**

Top N/Bottom N Search Rule

```
:CHSCanner:SWEep:TOPBottom:RULE SPEed|ACCUracy
:CHSCanner:SWEep:TOPBottom:RULE?
```

This command toggles the top/bottom N measurement rule between speed and accuracy. It's only available for Top N and Bottom N sweep mode.

Example: `:CHSCanner:SWEep:TOPBottom:RULE ACCUracy`

Key access: **Meas > Setup > Search Rule**

Top Cycle

```
:CHSCanner:SWEep:TOP:CYCLe[:STATe] OFF|ON|0|1
:CHSCanner:SWEep:TOP:CYCLe[:STATe]?
```

Use this command to turn on/off Top Cycle state. It's only available for Top N sweep mode.

```
:CHSCanner:SWEep:TOP:CYCLe:VALue <value>
:CHSCanner:SWEep:TOP:CYCLe:VALue?
```

This command set the Top Cycle value.

Range: 1 to 10000

Example: `:CHSCanner:SWEep:TOP:CYCLe:VALue 30`

Key access: **Meas > Setup > Top Cycle**

Bottom Cycle

```
:CHSCanner:SWEep:BOTTom:CYCLe[:STATe] OFF|ON|0|1
:CHSCanner:SWEep:BOTTom:CYCLe[:STATe]?
```

Use this command to turn on/off Bottom Cycle state. It's only available for Bottom N sweep mode.

```
:CHSCanner:SWEep:BOTTom:CYCLe:VALue <value>
:CHSCanner:SWEep:BOTTom:CYCLe:VALue?
```

Use this command to define the Bottom Cycle value. It's only available for Bottom N sweep mode.

Range: 1 to 10000

Example: `:CHSCanner:SWEep:BOTTom:CYCLe:VALue 10`

Key access: **Meas > Setup > Bottom Cycle**

Scan Start/Stop State Switch

```
:CHSCanner:SWEep[:STATe] RUN|STOP
:CHSCanner:SWEep[:STATe]?
```

This command toggles the scan state between start and stop.

Example: `:CHSCanner:SWEep RUN`

Key access: **Meas > Scan Start**

Channel Scanner Edit Subsection

Set Channel Edit Method for List Sweep

```
:CHSCanner:EDIT:LIST:METHod
<index>,<0|1>{,<index>,< 0|1>}
:CHSCanner:EDIT:LIST:METHod?
```

This command specify the channel edit method for list sweep. In list sweep mode, there are two method to insert or edit a scan channel, custom method and signal standard method. Custom method requires to insert or edit a scan channel by inputting the center frequency and bandwidth of channel, and signal standard method requires to choose the signal standard and channel ID from built-in signal standard list.

Remarks: Parameter <0|1> indicate <Custom | Standard>

Example: `:CHSCanner:EDIT:LIST:METHod 1,0,3,1`

Key access: **Meas > List Edit > Insert > Edit Method**

Add Channel By Custom For List Sweep

```
:CHSCanner:EDIT:LIST:ADD:CUSTom
<freq>,<bandwidth>{,<freq>,<bandwidth>}
```

This command adds custom scan channels to the sweep list of List Sweep Mode.

Example:
```
:CHSCanner:EDIT:LIST:ADD:CUSTom
10000000,1000000,20000000,10000
```

Key access: **Meas > List Edit > Insert > Edit Method > Custom**

Add Channel By Signal Standard for List Sweep

```
:CHSCanner:EDIT:LIST:ADD:STD <signal std.>,<ch.
no>{,<signal std.>,<ch. no>}
```

This command adds scan channels according to the build-in signal standard. The address of build-in signal stand file is “d:/”.

Example:
```
:CHSCanner:EDIT:LIST:ADD:STD
"d:/CDMA-AWS_DL.STD,11"
```

Key access: **Meas > List Edit > Insert > Edit Method Std > Signal Std**

Modify Channel By Custom For List Sweep

```
:CHSCanner:EDIT:LIST:MODify:CUSTom
<index>,<freq>,<band-
width>{,<index>,<freq>,<bandwidth>}
```

This command edit the sweep channel by custom, and it requires to input the channel index, new center frequency and new bandwidth.

Example:
```
:CHSCanner:EDIT:LIST:MODify:CUSTom
1,10000000,10000,2,20000000,20000
```

Key access: **Meas > List Edit > Edit > Edit Method > Custom**

Modify Channel By Signal Standard for List Sweep

```
:CHSCanner:EDIT:LIST:MODify:STD <index>,<signal
std.>,<ch. no>{,<index>,<signal std.>,<ch. no>}
```

This command modify scan channels according to the build-in signal standard. The address of build-in signal stand file is “d:/”.

Example:
```
:CHSCanner:EDIT:LIST:MODify:STD
“1,d:/CDMA-AWS_DL.STD,13”
```

Key access: **Meas > List Edit > > Edit > Edit Method Std> Signal Std**

Delete Channel For List Sweep

```
:CHSCanner:EDIT:LIST:DELete <index>{,<index>}
```

This command deletes channels for list sweep mode.

Example:
```
:CHSCanner:EDIT:LIST:DELete 1
```

Key access: **Meas > List Edit > Delete**

Delete All Channels For List Sweep

```
:CHSCanner:EDIT:LIST:DELete:ALL
```

This command delete all channels for list sweep mode.

Example:
```
:CHSCanner:EDIT:LIST:DELete:ALL
```

Key access: **Meas > List Edit > Delete All**

Query Channel Information For List Sweep

```
:CHSCanner:EDIT:LIST?
```

This command query the scan channel information for list sweep.

Example: `:CHSCanner:EDIT:LIST?`

Query Custom Channel For List Sweep

```
:CHSCanner:EDIT:LIST:CUSTom?
```

This command query the custom channel information for list sweep.

Example: `:CHSCanner:EDIT:LIST:CUSTom?`

Query Signal Standard Channel for List Sweep

```
:CHSCanner:EDIT:LIST:STD?
```

This command query the signal standard channel for list sweep.

Example: `:CHSCanner:EDIT:LIST:STD?`

Set Edit Method For Top N Sweep

```
:CHSCanner:EDIT:TOP:METHod STD|CUSTom
:CHSCanner:EDIT:TOP:METHod?
```

This command specify the edit method for Top N sweep mode.

Example: `:CHSCanner:EDIT:TOP:METHod CUST`

Key access: **Meas > Scan Mode > Top N > Return > Range Edit > Edit Method**

Set Scan Channel For Top N Sweep By Signal Standard

```
:CHSCanner:EDIT:TOP:STD <value>
:CHSCanner:EDIT:TOP:STD?
```

This command defines the signal standard for Top N sweep method.

Example: `:CHSCanner:EDIT:TOP:STD "d:/CDMA-AWS_DL.STD"`

Key access: **Meas > Range Edit > Edit Method Std > Signal Std**

Set Scan Channel Range For Top N Sweep By Signal Standard

```
:CHSCanner:EDIT:TOP:STD:RANGe <value>,<value>
:CHSCanner:EDIT:TOP:STD:RANGe?
```

This command specify scan channel range number for Top N sweep mode. The signal standard should defined before this operation.

Example: `:CHSCanner:EDIT:TOP:STD:RANGe 10,100`

Key access: **Meas > Range Edit > Edit Method Std > Signal Std, then >Begin Chn, > End Chn**

Set Custom Scan Channel For Top N Sweep

```
:CHSCanner:EDIT:TOP:CUSTom <start freq>,<freq
step>,<bandwidth>,<ch. number>
:CHSCanner:EDIT:TOP:CUSTom?
```

This command specify the customer scan channels for Top N Sweep. Need to input the start frequency, bandwidth of each channel and channel number.

Example: `:CHSCanner:EDIT:TOP:CUSTom 10000000,1000000,10000,100`

Key access: **Meas > Range Edit > Edit Method Custom**

Set Start Frequency For Custom Scan Channel For Top N Sweep

```
:CHSCanner:EDIT:TOP:CUSTom:STARt <value>
:CHSCanner:EDIT:TOP:CUSTom:STARt?
```

This command specify the start frequency of custom scan channels for Top N sweep mode.

Example: `:CHSCanner:EDIT:TOP:CUSTom:STARt 20000000`

Key access: **Meas > Range Edit > Edit Method Custom, then > Start Freq**

Set Frequency Step for Customer Scan Channel For Top N Sweep

```
:CHSCanner:EDIT:TOP:CUSTom:STEP <value>
:CHSCanner:EDIT:TOP:CUSTom:STEP?
```

This command specify the frequency step size of custom scan channels for Top N sweep mode.

Example: `:CHSCanner:EDIT:TOP:CUSTom:STEP 2000000`

Key access: **Meas > Range Edit > Edit Method Custom, then > Freq Step Size**

Set Bandwidth For Custom Scan Channel For Top N Sweep

```
:CHSCanner:EDIT:TOP:CUSTom:BANDwidth <value>
:CHSCanner:EDIT:TOP:CUSTom:BANDwidth?
```

This command specify the bandwidth of custom scan channels for Top N sweep mode.

Example: `:CHSCanner:EDIT:TOP:CUSTom:BANDwidth 30000`

Key access: **Meas > Range Edit > Edit Method Custom, then > Bandwidth**

Set Custom Scan Channel Quantity For Top N Sweep

```
:CHSCanner:EDIT:TOP:CUSTom:CHNumber <value>
:CHSCanner:EDIT:TOP:CUSTom:CHNumber?
```

This command defines custom scan channel quantity for Top N sweep mode.

Example: `:CHSCanner:EDIT:TOP:CUSTom:CHNumber 200`

Key access: **Meas > Range Edit > Edit Method Custom, then > Number of Chn**

Set Edit Method For Bottom N Sweep

```
:CHSCanner:EDIT:BOTTom:METHod STD|CUSTom
:CHSCanner:EDIT:BOTTom:METHod?
```

This command specify the edit method for Bottom N sweep mode.

Example: `:CHSCanner:EDIT:BOTTom:METHod CUST`

Key access: **Meas > Scan Mode > Bottom N > Return > Range Edit > Edit Method**

Set Scan Channel For Bottom N Sweep By Signal Standard

```
:CHSCanner:EDIT:BOTTom:STD <value>
:CHSCanner:EDIT:BOTTom:STD?
```

Use this command to specify the signal standard for Bottom N sweep method.

Example: `:CHSCanner:EDIT:BOTTom:STD "d:/CDMA-AWS_DL.STD"`

Key access: **Meas > Range Edit > Edit Method Std > Signal Std.**

Set Scan Channel Range For Bottom N Sweep By Signal Standard

```
:CHSCanner:EDIT:BOTTom:STD:RANGe <value>,<value>
:CHSCanner:EDIT:BOTTom:STD:RANGe?
```

This command specify scan channel range number for Bottom N sweep mode. The signal standard should defined before this operation.

Example: `:CHSCanner:EDIT:BOTTom:STD:RANGe 10,100`

Key access: **Meas > Range Edit > Edit Method Std > Signal Std, then >Begin Chn, > End Chn**

Set Custom Scan Channel For Bottom N Sweep

```
:CHSCanner:EDIT:BOTTom:CUSTom <start freq>,<freq
step>,<bandwidth>,<ch. number>
:CHSCanner:EDIT:BOTTom:CUSTom?
```

This command specify the customer scan channels for Bottom N sweep mode.

Example:
```
:CHSCanner:EDIT:BOTTom:CUSTom
10000000,1000000,10000,100
```

Key access: **Meas > Range Edit > Edit Method Custom**

Set Start Frequency For Custom Scan Channel For Bottom N Sweep

```
:CHSCanner:EDIT:BOTTom:CUSTom:STARt <value>
:CHSCanner:EDIT:BOTTom:CUSTom:STARt?
```

This command specify the start frequency of custom scan channels for Bottom N sweep mode.

Example: `:CHSCanner:EDIT:BOTTom:CUSTom:STARt 20000000`

Key access: **Meas > Range Edit > Edit Method Custom, then > Start Freq**

Set Frequency Step for Customer Scan Channel For Bottom N Sweep

```
:CHSCanner:EDIT:BOTTom:CUSTom:STEP <value>
:CHSCanner:EDIT:BOTTom:CUSTom:STEP?
```

This command specify the frequency step size of custom scan channels for Bottom N sweep mode.

Example: `:CHSCanner:EDIT:BOTTom:CUSTom:STEP 2000000`

Key access: **Meas > Range Edit > Edit Method Custom, then > Freq Step Size**

Set Bandwidth For Custom Scan Channel For Bottom N Sweep

```
:CHSCanner:EDIT:BOTTom:CUSTom:BANDwidth <value>
:CHSCanner:EDIT:BOTTom:CUSTom:BANDwidth?
```

This command specify the bandwidth of custom scan channels for Bottom N sweep mode.

Example: `:CHSCanner:EDIT:BOTTom:CUSTom:BANDwidth 30000`

Key access: **Meas > Range Edit > Edit Method Custom, then > Bandwidth**

Set Custom Scan Channel Quantity For Bottom N Sweep

```
:CHSCanner:EDIT:BOTTom:CUSTom:CHNumber <value>
:CHSCanner:EDIT:BOTTom:CUSTom:CHNumber?
```

This command defines custom scan channel quantity for Bottom N sweep mode.

Example: `:CHSCanner:EDIT:BOTTom:CUSTom:CHNumber 200`

Key access: **Meas > Range Edit > Edit Method Custom, then > Number of Chn**

Channel Scanner Display Subsection

Display Switch

```
:CHSCanner:DISPlay[:STATe] RUN|STOP
:CHSCanner:DISPlay[:STATe]?
```

This command toggles channel scanner display state between freeze and refresh. Freeze state stops channel scanning operation, and Refresh state runs channel scanning operation.

Example: `:CHSCanner:DISPlay RUN`

Key access: **Shift > Disp > Freeze/Refresh**

Display Mode

```
:CHSCanner:DISPlay:MODE TABLe|CHARt
:CHSCanner:DISPlay:MODE?
```

This command toggles display mode between table mode and time chart mode. The time chart mode is only available for List sweep mode.

Example: `:CHSCanner:DISPlay:MODE CHARt`

Key access: **Shift > Disp > Disp Mode**

Display Threshold State

```
:CHSCanner:DISPlay:THReshold [:STATe] OFF|ON|0|1
:CHSCanner:DISPlay:THReshold [:STATe]?
```

This command toggles display threshold state. Then the display threshold is turned on, only the channels with higher channel power than the threshold value will be displayed on the screen.

Remark: This parameter is only available for Table (Bar) display mode.

Example:
```
:CHSCanner:DISPlay:THREShold 1
```

Key access: **Shift > Disp > Disp Mode > Threshold**

Display Threshold Value

```
:CHSCanner:DISPlay:THReshold:VALue <value>
:CHSCanner:DISPlay:THReshold:VALue?
```

This command defines the display threshold value.

Example:
```
:CHSCanner:DISPlay:THReshold:VALue -70
```

Key access: **Shift > Disp > Disp Mode > Threshold**

Limit State

```
:CHSCanner:DISPlay:LIMit[:STATe] OFF|ON|0|1
:CHSCanner:DISPlay:LIMit[:STATe]?
```

Use this command to turn on/off limit state in channel scanner mode.

Example:
```
:CHSCanner:DISPlay:LIMit ON
```

Key access: **Shift > Limit > Limit**

Limit Value

```
:CHSCanner:DISPlay:LIMit:VALue <value>
:CHSCanner:DISPlay:LIMit:VALue?
```

This command specify the limit value in channel scanner mode.

Example:
```
:CHSCanner:DISPlay:LIMit:VALue -20
```

Key access: **Shift > Limit > Limit**

Limit Type

```
:CHSCanner:DISPlay:LIMit:TYPE UPPer|LOWer
:CHSCanner:DISPlay:LIMit:TYPE?
```

This command toggles limit type between upper and lower.

Example: `:CHSCanner:DISPlay:LIMit:TYPE LOWer`

Key access: **Shift > Limit > Limit Type**

Display Sort Type

```
:CHSCanner:DISPlay:SORT ID|POWer
:CHSCanner:DISPlay:SORT?
```

This command toggles display sort type between by ID and by power.

Example: `:CHSCanner:DISPlay:SORT POWer`

Key access: **Shift > Disp > Sort By**

Display Sort Order

```
:CHSCanner:DISPlay:SORT:ORDer DOWN|UP
:CHSCanner:DISPlay:SORT:ORDer?
```

This command toggle display sort order between down and up.

Example: `:CHSCanner:DISPlay:SORT:ORDer UP`

Key access: **Shift > Disp > Sort Order**

Channel Scanner Logging Subsection

Logging State

```
:CHSCanner:LOG[:STATe] RUN|STOP
```

Use this command to start/stop long time logging of channel scanner measurement result.

Example: `:CHSCanner:LOG RUN`

Key access: **Meas > Logging Start**

Logging File Type

```
:CHSCanner:LOG:TYPE CSV|KML
:CHSCanner:LOG:TYPE?
```

This command switch the logging file type between CSV and KML.

Example: `:CHSCanner:LOG:TYPE KML`

Key access: **Meas > Logging Start**

Channel Scanner Measure Subsection

Query Result

```
:CHSCanner:MEASure?
```

Use this command to query the channel scanner measurement result.

Example: `:CHSCanner:MEASure?`

Demodulation Option Subsystem

The Demodulation Option Subsystem provides you the SCPI command reference for the demodulation function. Use the command `:INSTrument MA` to access this subsystem.

Option AMA is required to perform AM and FM demodulation analysis, and option DMA is required to perform ASK and FSK demodulation analysis

Switch Demodulation Mode

```
:INSTrument:MEASure AM|FM|ASK|FSK
:INSTrument:MEASure?
```

After access to demodulation analysis mode, use this command to switch the 4 modulaes in demodulation mode.

Range: AM, FM, ASK,FSK

Example: `:INSTrument:MEASure AM`

Key access: **Mode > Modulation Analysis > AM**

Carrier Frequency

```
[:SENSe]:FREQuency:CARRier <freq>
[:SENSe]:FREQuency:CARRier?
```

This command sets the carrier frequency of the demodulation analysis.

Range: 2.5 MHz to 7 GHz

Example: `:FREQuency:CARRier 1 GHZ`

Key access: **Freq > Carrier Freq**

Carrier Frequency Step

```
[:SENSe]:FREQuency:CARRier:STEP <freq>
[:SENSe]:FREQuency:CARRier:STEP?
```

This command sets the carrier frequency step of the demodulation analysis.

Example: `:FREQuency:CARRier:STEP 50MHz`

Key access: **Freq > CarrFreq Step**

Auto Carrier Frequency State

```
[:SENSe]:FREQuency:CARRier:AUTO OFF|ON|1|0
[:SENSe]:FREQuency:CARRier:AUTO?
```

Use this command to turn on/off auto carrier frequency detection.

Example: `:FREQuency:CARRier:AUTO ON`

Key access: **Freq > Auto CarrFreq**

Attenuation

```
[:SENSe]:POWer[:RF]:ATTenuation <value>
[:SENSe]:POWer[:RF]:ATTenuation?
```

This command sets the RF attenuation value.

Range: 0 to 51 dB

Example: `:POWer:ATTenuation 20dB`

```
[:SENSe]:POWer[:RF]:ATTenuation:AUTO OFF|ON|0|1
[:SENSe]:POWer[:RF]:ATTenuation:AUTO?
```

Use this command to turn on/off auto attenuation status.

Example: `:POWer:ATTenuation:AUTO ON`

Key access: **Amptd > Attenuation**

Preamp

```
[:SENSe]:POWer[:RF]:GAIN[:STATe] OFF|ON|0|1
[:SENSe]:POWer[:RF]:GAIN[:STATe]?
```

Use this command to turn on/off preamplifier in demodulation mode.

Example: `:POWer:GAIN ON`

Key access: **Amptd > Preamp**

Single Sweep

```
:INITiate[:IMMediate]
```

Use this command to perform a single sweep (measurement).

Example: `:INITiate`

Key access: **Shift > Sweep > Single**

Sweep State

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

This command toggles the sweep state between single sweep and continuous sweep.

Example: `:INITiate:CONTinuous 1`

Key access: **Shift > Sweep > Sweep**

External Gain

```
[:SENSe]:CORRection:OFFSet[:MAGNitude] <dB>
[:SENSe]:CORRection:OFFSet[:MAGNitude]?
```

This command sets the external gain.

Range: 0 to 51 dB

Example: `:CORRection:OFFSet 20`

Key access: **Amptd > External Gain**

Restart Meas

```
[:SENSe]:MEASure:RESTart
```

Use this command to restart measurement of current demodulation analysis.

Example: `:MEASure:RESTart`

Key access: **Meas > Restart Meas**

Read Measurement Data

```
:CALCulate:DATA?
```

Use this command to query the measurement result of current demodulation analysis.

Example: `:CALCulate:DATA?`

AM Demodulation Subsection

Switch To AM

```
:INSTrument:MEASure AM
```

Use this command to switch to AM demodulation analysis.

Read AM Depth

```
:CALCulate:AM:MDEPth?
```

Use this command to query the AM depth measurement result.

Example: `:CALCulate:AM:MDEPth?`

Read Current AM Carrier Power

```
:CALCulate:AM:MDEPth?
```

Use this command to query the AM carrier power measurement result

Example: `:CALCulate:AM:MDEPth?`

Read Current AM Modulation Rate

```
:CALCulate:AMA:MRATe?
```

Use this command to query the AM Modulation rate result.

Example: `:CALCulate:AMA:MRATe?`

AM Average Count

```
[:SENSe]:AMA:AVERage:COUNt <integer>
[:SENSe]:AMA:AVERage:COUNt?
```

Use this command to set average count number in AM or FM demodulation analysis.

Example: `:AMA:AVERage:COUNt 50`

```
[:SENSe]:AMA:AVERage OFF|ON|1|0
[:SENSe]:AMA:AVERage?
```

Use this command to turn on/off average state in AM or FM demodulation analysis.

Example: `:AMA:AVERage 1`

Key access: **Meas > Average Number**

AM Detector

```
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]
PPK|NPK|PNPK|RMS
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]?
```

Use this command to switch the detector type in AM demodulation analysis.

Range: PPK, NPK, PNPK, RMS

Example: `:AMA:DEMod:DETector PPK`

Key access: **Meas > Detector**

AM Detector PeakHold

```
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]:PEAKhold
OFF|ON|0|1
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]:PEAKhold?
```

Use this command to turn on/off Peak hold state of AM detector.

Example: `:AMA:DEMod:DETector:PEAKhold ON`

Key access: **Meas > Detector > PeakHold**

AM IF Bandwidth

```
[:SENSe]:AMA:IFBWidth
1.2MHz|960kHz|600kHz|480kHz|300kHz|240kHz|120kHz
|96kHz|60kHz
[:SENSe]:AMA:IFBWidth?
```

This command sets the IF bandwidth of AM demodulation analysis.

Range: 1.2 MHz, 960 kHz, 600 kHz, 480 kHz, 300 kHz, 240 kHz, 120 kHz, 96 kHz, 60 kHz

Example: `:AMA:IFBWidth 96kHz`

Key access: **Meas > IFBW**

AM IF Bandwidth

```
[:SENSe]:AMA:IFBWidth:AUTO OFF|ON|0|1
```

Use this command to turn on/off the IF bandwidth auto state in AM demodulation mode.

Example: `:AMA:IFBWidth:AUTO ON`

Key access: **Meas > IFBW Auto/Man**

AM Equal Low Pass Filter

```
[:SENSe]:AMA:EQLPfilter
AUTO|OFF|IFBW6|IFBW20|IFBW60|IFBW200|IFBW600|IFBW2000
[:SENSe]:AMA:EQLPfilter?
```

Use this command to set the equal low pass filter in AM demodulation analysis.

Range: AUTO, OFF, IFBW6, IFBW20, IFBW60, IFBW200, IFBW600, IFBW2000

Example: `:AMA:EQLPfilter IFBW6`

Key access: **Meas > Equal LPF**

AM Limit State

```
:CALCulate:AMA:LIMit OFF|ON|0|1
:CALCulate:AMA:LIMit?
```

Use this command to turn on/off the limit in AM demodulation analysis.

Example: `:CALCulate:AMA:LIMit 1`

Key access: **Shift > Limit> Limit On/Off**

AM Carrier Power Upper Limit

```
:CALCulate:AMA:LIMit:POWer:UPPer <value>
:CALCulate:AMA:LIMit:POWer:UPPer?
```

Use this command to set the upper limit of AM carrier power.

Range: –100 to 30 dBm

Example: `:CALCulate:AMA:LIMit:POWer:UPPer -20`

Key access: **Shift > Limit> CarrPwr Upper**

AM Depth Upper Limit

```
:CALCulate:AM:LIMit:MDEPth:UPPer <value>
:CALCulate:AM:LIMit:MDEPth:UPPer?
```

Use this command to set the upper limit of AM depth.

Range: 0.1% to 100%

Example: `:CALCulate:AM:LIMit:MDEPth:UPPer 50`

Key access: **Shift > Limit> AMod Depth Up**

AM Depth Lower Limit

```
:CALCulate:AM:LIMit:MDEPth:LOWer <value>
:CALCulate:AM:LIMit:MDEPth:LOWer?
```

Use this command to set the lower limit of AM depth.

Range: 0.1% to 100%

Example: `:CALCulate:AM:LIMit:MDEPth:LOWer 10`

Key access: **Shift > Limit > AMod Depth Low**

AM Carrier Frequency Offset Upper Limit

```
:CALCulate:AMA:LIMit:FOFFset:UPPer <freq>
:CALCulate:AMA:LIMit:FOFFset:UPPer?
```

Use this command to turn on/off peak hold state of AM detector.

Range: –1MHz to +1MHz

Example: `:CALCulate:AMA:LIMit:FOFFset:UPPer 1kHz`

Key access: **Shift > Limit > CarrFreqOfStUp**

AM X Scale/DIV

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion < value >
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion?
```

Use this command to set the Scale per division of X axis. The unit of this parameter is second. This command is only available when the scale coupling state is set to off.

Range: 1e–9 to 1

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:PDIVision 2e–4`

Key access: **Shift > Disp >X Scale > Scale/DIV**

AM X Reference Value

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue
< value >
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue?
```

Use this command to set the reference value of X axis.

Range: –5 to 5 s

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:RVALue 1`

Key access: **Shift > Disp >X Scale > Ref Value**

AM X Reference Position

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSi-
tion
LEFT|CENTer|RIGHt
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSi-
tion?
```

Use this command to set the reference position of X axis.

Range: LEFT, CENTer, RIGHt

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:RPOSition CENT`

Key access: **Shift > Disp >X Scale > Ref Position**

AM X Scale Coupling State

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle
0|1|OFF|ON
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle?
```

Use this command to turn on/off scale coupling state.

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:COUPle ON`

Key access: **Shift > Disp >X Scale > Scale Coupling**

AM Y Reference Value

```
:DISPlay:AM:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue <
value>
:DISPlay:AM:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue?
```

Use this command to set reference value of Y axis. The unit of this parameter is %.

Range: –150% to 150%

Example: `:DISPlay:AM:VIEW:WINDow:TRACe:Y:RVALue 20`

Key access: **Shift > Disp >Y Scale > Ref Value**

AM Y Scale/DIV

```
:DISPlay:AM:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVi-
sion < double >
:DISPlay:AM:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVi-
sion?
```

Use this command to set the Scale per division of Y axis. The unit of this parameter is %.

Range: 0.1% to 100%

Example: `:DISPlay:AM:VIEW:WINDow:TRACe:Y:PDIVision 50`

Key access: **Shift > Disp >Y Scale > Scale/DIV**

AM Y Ref Position

```
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSi-
tion TOP|CENTer|BOTTom
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSi-
tion?
```

Use this command to set the reference position of Y axis.

Range: TOP, CENTer, BOTTom

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:Y:RPOSition TOP`

Key access: **Shift > Disp >Y Scale > Ref Position**

AM Y Scale Coupling State

```
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:COUPle
0|1|OFF|ON
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:COUPle?
```

Use this command to turn on/off Y scale coupling state.

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:Y:COUPle 1`

Key access: **Shift > Disp >Y Scale > Scale Coupling**

FM Demodulation Subsection

Switch to FM

```
:INSTrument:MEASure FM
```

Use this command to switch to FM demodulation analysis.

Query FM Carrier Power

```
:CALCulate:AMA:POWer?
```

Use this command to query FM carrier power measurement result.

Example: `:CALCulate:AMA:POWer?`

Query FM Modulation Rate

```
:CALCulate:AMA:MRATe?
```

Use this command to query FM modulation rate measurement result in FM demodulation analysis.

Example: `:CALCulate:AMA:MRATe?`

Query FM Frequency Deviation

```
:CALCulate:FM:FDEViation?
```

Use this command to query FM frequency deviation test result.

Example: `:CALCulate:FM:FDEViation?`

FM Average Count

```
[:SENSe]:AMA:AVERage:COUNt <value>
[:SENSe]:AMA:AVERage:COUNt?
```

This command sets average count number in FM demodulation analysis.

Example: `:AMA:AVERage:COUNt 50`

Key access: **Meas > Average Number**

FM Average Count State

```
[:SENSe]:AMA:AVERage OFF|ON|1|0
```

```
[:SENSe]:AMA:AVERage?
```

This command turns on/off average state in the FM demodulation analysis.

Example: `:AMA:AVERage 1`

Key access: **Meas > Average Number**

FM Detector

```
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]
PPK|NPK|PNPK|RMS
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]?
```

Use this command to specify the detector in FM demodulation analysis.

Example: `:AMA:DEMod:DETector PPK`

Key access: **Meas > Detector**

FM Detector PeakHold

```
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]:PEAKhold
OFF|ON|0|1
[:SENSe]:AMA:DEMod:DETector[:FUNCtion]:PEAKhold?
```

Use this command to turn on/off peak hold state of detector.

Example: `:AMA:DEMod:DETector:PEAKhold ON`

Key access: **Meas > Detector > PeakHold**

FM IF Bandwidth

```
:SENSe]:AMA:IFBWidth
1.2MHz|960kHz|600kHz|480kHz|300kHz|240kHz|120kHz
|96kHz|60kHz
[:SENSe]:AMA:IFBWidth?
```

This command specifies IF bandwidth of FM demodulation analysis.

Range: 1.2 MHz, 960 kHz, 600 kHz, 480 kHz, 300 kHz, 240 kHz, 120 kHz, 96 kHz, 60 kHz

Example: `:AMA:IFBWidth 96kHz`

```
[:SENSe]:AMA:IFBWidth:AUTO OFF|ON|0|1
[:SENSe]:AMA:IFBWidth:AUTO?
```

This command turns on/off IF bandwidth of IF demodulation analysis.

Example: `:AMA:IFBWidth:AUTO ON`

Key access: **Meas > IFBW**

FM Equal Low Pass Filter

```
[:SENSe]:AMA:EQLPfilter
AUTO|OFF|IFBW6|IFBW20|IFBW60|IFBW200|IFBW600|IFBW2000
[:SENSe]:AMA:EQLPfilter?
```

This command specifies the equal low pass filter type.

Range: AUTO, OFF, IFBW6, IFBW20, IFBW60, IFBW200, IFBW600, IFBW2000

Example: `:AMA:EQLPfilter IFBW6`

Key access: **Meas > Equal LPF**

FM Limit State

```
:CALCulate:AMA:LIMit OFF|ON|0|1
:CALCulate:AMA:LIMit?
```

Use this command to turn on/off limit state of FM demodulation analysis.

Example: `:CALCulate:AMA:LIMit 1`

Key access: **Shift > Limit > Limit**

FM Carrier Power Upper Limit

```
:CALCulate:AMA:LIMit:POWer:UPPer <value>
:CALCulate:AMA:LIMit:POWer:UPPer?
```

Use this command to specify the carrier power upper limit of FM demodulation analysis.

Range: –100 to 30 dBm

Example: `:CALCulate:AMA:LIMit:POWer:UPPer –50`

Key access: **Shift > Limit > CarrPwr Upper**

FM Frequency Deviation Upper Limit

```
:CALCulate:FM:LIMit:FDUL:UPPer <freq>
:CALCulate:FM:LIMit:FDUL:UPPer?
```

Use this command to specify frequency deviation upper limit of FM demodulation analysis.

Range: 1 Hz to 500 kHz

Example: `:CALCulate:FM:LIMit:FDUL:UPPer 2kHz`

Key access: **Shift > Limit > FreqDev Upper**

FM frequency Deviation Lower Limit

```
:CALCulate:FM:LIMit:FDLL:LOWer <freq>
:CALCulate:FM:LIMit:FDLL:LOWer?
```

Use this command to specify frequency deviation lower limit of FM demodulation analysis.

Range: 1 Hz to 500 kHz

Example: `:DANalyse:FM:LIMit:FDLL:LOWer 3kHz`

Key access: **Shift > Limit > FreqDev Lower**

FM Carrier Frequency Offset Upper Limit

```
:CALCulate:AMA:LIMit:FOFFset:UPPer <freq>
:CALCulate:AMA:LIMit:FOFFset:UPPer?
```

Use this command to specify FM carrier frequency offset upper limit of FM demodulation analysis.

Range: `–1 to 1 MHz`

Example: `:CALCulate:AMA:LIMit:FOFFset:UPPer 0`

Key access: **Shift > Limit > CarrFreqOfstUp**

FM X Scale/DIV

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion <real_time>
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion?
```

This command specifies scale per division of X axis.

Range: 1e–1 to 1

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:PDIVision 1e–8`

Key access: **Shift > Disp > X Scale > Scale/DIV**

FM X Reference Value

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue
< real_time >
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe
]:RVALue?
```

This command specifies the reference value of X axis.

Range: –5 to +5

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:RVALue 1e–8`

Key access: **Shift > Disp > X Scale > Ref Value**

FM X Reference Position

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSiti
on LEFT|CENTer|RIGHt
```

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSiti
on?
```

This command specifies the reference position of X axis.

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:RPOSition LEFT`

Key access: **Shift > Disp > X Scale > Ref Position**

FM X Scale Coupling State

```
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle
0|1|OFF|ON
:DISPlay:AMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle?
```

Use this command to turn on/off scale coupling state of X axis.

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:X:COUPle ON`

Key access: **Shift > X Scale > Scale Coupling**

FM Y Reference Value

```
:DISPlay:FM:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue
<freq >
:DISPlay:FM:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue?
```

Use this command to specify reference value of Y axis.

Range: –1 to 1 MHz

Example: `:DISPlay:FM:VIEW:WINDow:TRACe:Y:RVALue 10kHz`

Key access: **Shift > Disp > Y Scale > Ref Value.**

FM Y Scale/DIV

```
:DISPlay:FM:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVi-
sion <freq >
:DISPlay:FM:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVi-
sion?
```

Use this command to specify scale per division of Y axis.

Range: 1 Hz to 100 kHz

Example: `:DISPlay:FM:VIEW:WINDow:TRACe:Y:PDIVision 20kHz`

Key access: **Shift > Disp > Y Scale > Ref Value**

FM Y Reference Position

```
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSi-
tion TOP|CENTer|BOTTom
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSi-
tion?
```

Use this command to specify reference position of Y axis.

Range: TOP, CENTer, BOTTom

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:Y:RPOSition TOP`

Key access: **Shift > Disp > Y Scale > Ref Position**

FM Y Scale Coupling State

```
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:COUPle
0|1|OFF|ON
:DISPlay:AMA:VIEW:WINDow:TRACe:Y[:SCALe]:COUPle?
```

Use this command to turn on/off scale coupling state of Y axis.

Example: `:DISPlay:AMA:VIEW:WINDow:TRACe:Y:COUPle 1`

Key access: **Shift > Disp > Y Scale > Scale Coupling**

ASK Demodulation Subsection

Switch To ASK

```
:INSTrument:MEASure ASK
```

Use this command to switch to ASK demodulation analysis.

ASK Average Count State

```
[:SENSe]:DMA:AVERage OFF|ON|0|1
[:SENSe]:DMA:AVERage?
```

Use this command to turn on/of average operation.

Example: `:DMA:AVERage 1`

Key access: **Meas > Average Number**

ASK Average Count

```
[:SENSe]:DMA:AVERage:COUNt <value>
[:SENSe]:DMA:AVERage:COUNt?
```

Use this command to specify average count number in ASK demodulation analysis.

Example: `:DMA:AVERage:COUNt 20`

Key access: **Meas > Average Number**

ASK Symbol Rate

```
[:SENSe]:DMA:RADio:STANdard:SRATe <value>
[:SENSe]:DMA:RADio:STANdard:SRATe?
```

Use this command to specify symbol rate for ASK demodulation analysis.

Range: 100 sps to 100 ksps

Example: `:DMA:RADio:STANdard:SRATe 2ksps`

Key access: **Meas > Symbol Rate**

ASK Measurement Filter

```
[:SENSe]:DMA:RADio:STANdard:FILTer:MEASurement
NONE|RNYQuist|NYQuist|GAUSsian
[:SENSe]:DMA:RADio:STANdard:FILTer:MEASurement?
```

Use this command to set the measurement filter in ASK demodulation analysis.

Range: RNYQuist, NYQuist, GAUSsian, NONE

Example: `:DMA:RADio:STANdard:FILTer:MEASurement RNYQuist`

Key access: **Meas > Filter Setup > Meas Filter**

ASK Reference Filter

```
[:SENSe]:DMA:RADio:STANdard:FILTer:REFerence
RNYQuist|NYQuist|GAUSsian|NONE
[:SENSe]:DMA:RADio:STANdard:FILTer:REFerence?
```

Use this command to specify reference filter in ASK demodulation analysis.

Range: RNYQuist, NYQuist, GAUSsian, NONE

Example: `:DMA:RADio:STANdard:FILTer:REFerence RNYQuist`

Key access: **Meas > Filter Setup > Ref Filter**

ASK Filter Symbols

```
[:SENSe]:DMA:RADio:STANdard:FILTer:SYMBol <
integer >
[:SENSe]:DMA:RADio:STANdard:FILTer:SYMBol?
```

Use this command to specify ASK filter symbols.

Range: 2 to 25

Example: `:DMA:RADio:STANdard:FILTer:SYMBol 4`

Key access: **Meas > Filter Setup > Filter Symbols**

ASK Alpha/BbT

```
[:SENSe]:DMA:RADio:STANdard:ALPHa <real>
[:SENSe]:DMA:RADio:STANdard:ALPHa?
```

Use this command to specify the filter Alpha/Bbt value.

Range: 0.2 to 1

Example: `:DMA:RADio:STANdard:ALPHa 0.2`

Key access: **Meas > Filter Setup > Alpha/BbT**

ASK Measurement Length

```
[:SENSe]:DMA:MEASurement:LENGth <value>
[:SENSe]:DMA:MEASurement:LENGth?
```

Use this command to specify measurement length of ASK demodulation analysis.

Range: 20 to 1100

Example: `:DMA:MEASurement:LENGth 200`

Key access: **Meas > Meas Length**

ASK Trigger Type

```
:TRIGger:DMA[:SEQuence]:SOURce IMMediate|EXTer-
nal|RFTRigger
:TRIGger:DMA[:SEQuence]:SOURce?
```

Use this command to specify the trigger type used in current ASK demodulation analysis.

Range: IMMediate, EXTernal, RFTRigger

Example: `:TRIGger:DMA:SOURce EXTernal`

Key access: **Meas > Trigger**

ASK RF Trigger Level

```
:TRIGger:DMA[:SEQuence]:LEVel:ABSolute <value>
:TRIGger:DMA[:SEQuence]:LEVel:ABSolute?
```

Use this command to turn on/off scale coupling state of X axis.

Range: `–40 to 10 dBm`

Example: `:TRIGger:DMA:LEVel:ABSolute 5dBm`

Key access: **Meas > Trigger > RF Trigger > Video Level**

ASK External Trigger Type

```
:TRIGger:DMA[:SEQuence]:EXTernal:SLOPe POSi-
tive|NEGative
:TRIGger:DMA[:SEQuence]:EXTernal:SLOPe?
```

This command switch external trigger type between rise slope and fall slope. It's only available for external trigger.

Range: POSitive, NEGative

Example: `:TRIGger:DMA:EXTernal:SLOPe NEGative`

Key access: **Meas > Trigger >External > Trigger Slope**

ASK Trigger Delay Time

```
:TRIGger:DMA[:SEQuence]:DELay <time>
:TRIGger:DMA[:SEQuence]:DELay?
```

Use this command to specify trigger delay time.

Range: 0 to 200 s

Example: `:TRIGger:DMA:DELay 2us`

```
:TRIGger:DMA[:SEQuence]:DELay:STATe OFF|ON|0|1
:TRIGger:DMA[:SEQuence]:DELay:STATe?
```

Use this command to turn on/off trigger delay state.

Example: `:TRIGger:DMA:DELay:STATe ON`

Key access: **Meas > Trigger > External > Trig Delay**

ASK Limit State

```
:CALCulate:DMA:LIMit OFF|ON|0|1
:CALCulate:DMA:LIMit?
```

Use this command to turn on/off limit state in ASK demodulation analysis.

Example: `:CALCulate:DMA:LIMit ON`

Key access: **Shift > Limit > Limit**

ASK Carrier Power Upper Limit

```
:CALCulate:DMA:LIMit:POWer:UPPer <value>
:CALCulate:DMA:LIMit:POWer:UPPer?
```

Use this command to specify the carrier power upper limit value of ASK demodulation analysis.

Range: –100 to 30 dBm

Example: `:CALCulate:DMA:LIMit:POWer:UPPer 20dBm`

Key access: **Shift > Limit > CarrPwr Upper**

ASK Depth Upper Limit

```
:CALCulate:ASK:LIMit:AMDepth:UPPer <real>
:CALCulate:ASK:LIMit:AMDepth:UPPer?
```

Use this command to specify the depth upper limit value of ASK demodulation analysis.

Range: 0.1% to 100%

Example: `:CALCulate:ASK:LIMit:AMDepth:UPPer 20`

Key access: **Shift > Limit > ASK Depth Up**

ASK Depth Lower Limit

```
:CALCulate:ASK:LIMit:AMDepth:LOWer <real>
:CALCulate:ASK:LIMit:AMDepth:LOWer?
```

Use this command to specify the depth lower limit value of ASK demodulation analysis.

Range: 0.1% to 100%

Example: `:CALCulate:ASK:LIMit:AMDepth:LOWer 20`

Key access: **Shift > Limit > ASK Depth Lower Limit**

ASK Carrier Frequency Offset Upper Limit

```
:CALCulate:DMA:LIMit:FOFFset:UPPer <frequency>
:CALCulate:DMA:LIMit:FOFFset:UPPer?
```

Use this command to specify carrier frequency offset upper limit of ASK demodulation analysis.

Range: -2 to 2 MHz

Example: `:CALCulate:DMA:LIMit:FOFFset:UPPer 20 kHz`

Key access: **Shift > Limit > CarrFreqOfstUp**

ASK View Type

```
:DISPlay:DMA:VIEW WAVeform |SYMBol|EYE|ERRor
:DISPlay:DMA:VIEW?
```

Use this command to toggle the different view type of ASK demodulation analysis.

Range: SYMBol, WAVeform, EYE, ERRor

Example: `:DISPlay:DMA:VIEW SYMBol`

Key access: **Shift > Disp > Display**

ASK X Scale/DIV

```
DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion <value>
DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion?
```

Use this command to specify the scale per division of X axis of ASK demodulation analysis.

Range: 1 to 40

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:X:PDIVision 1`

Key access: **Shift > Disp > X Scale > Scale/DIV**

ASK X Reference Value

```
DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue
<integer>
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue?
```

Use this command to specify the reference value of X axis.

Range: –400 to 400

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:X:RVALue 2`

Key access: **Shift > Disp > X Scale > Ref Value**

ASK X Reference Position

```
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSi-
tion LEFT|CENTer|RIGHt
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSi-
tion?
```

Use this command to specify the reference position of X axis.

Range: LEFT, CENTer, RIGHt

Example: `:DANalyse:ASK:VIEW:X:RPOSition CENTer`

Key access: **Shift > Disp > X Scale > Ref Position**

ASK X Scale Coupling State

```
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle
0|1|OFF|ON
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle?
```

Use this command to turn on/off scale coupling state of X axis.

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:X:COUPle ON`

Key access: **Shift > X Scale > Scale Coupling**

ASK Y Reference Value

```
:DISPlay:ASK:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue
<real>
:DISPlay:ASK:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue?
```

Use this command to specify reference value of Y axis.

Range: –150% to 150%

Example: `:DISPlay:ASK:VIEW:WINDow:TRACe:Y:RVALue –1`

Key access: **Shift > Disp > Y Scale > Ref Value**

ASK Y Scale/DIV

```
DISPlay:ASK:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVi-
sion <real>
DISPlay:ASK:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVi-
sion?
```

Use this command to specify scale per division of Y axis.

Range: 0.1% to 100%

Example: `:DISPlay:ASK:VIEW:WINDow:TRACe:Y:PDIVision 1.2`

Key access: **Shift > Disp > Y Scale > Scale /DIV**

ASK Y Reference Position

```
:DISPlay:DMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSi-
tion TOP|CENTer|BOTTom
:DISPlay:DMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSi-
tion?
```

Use this command to specify the reference position of Y axis.

Range: TOP, CENTer, BOTTom

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:Y:RPOSition TOP`

Key access: **Shift > Disp > Y Scale > Ref Position**

ASK Y Scale Coupling State

```
:TRACe[:DATA]?
```

Use this command to query measurement trace data.

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:Y:COUPle 1`

Key access: **Shift > Disp > Y Scale > Scale Coupling**

Read ASK Trace Data

```
:DISPlay:FM:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue
<freq >
:DISPlay:FM:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue?
```

Use this command to specify reference value of Y axis.

Example: `:TRACe?`

FSK Demodulation Subsection

Switch To FSK

```
:INSTrument:MEASure FSK
```

Use this command to switch to FSK demodulation analysis.

FSK Average Count

```
[:SENSe]:DMA:AVERage OFF|ON|0|1
[:SENSe]:DMA:AVERage?
```

Use this command to turn on/of average operation.

Example: `:DMA:AVERage 1`

```
[:SENSe]:DMA:AVERage:COUNt <value>
```

```
[:SENSe]:DMA:AVERage:COUNt?
```

Use this command to specify average count number of FSK demodulation analysis.

Example: `:DMA:AVERage:COUNt 20`

Key access: **Meas > Average Number**

FSK Symbol Rate

```
[:SENSe]:DMA:RADio:STANdard:SRATe <sps>
[:SENSe]:DMA:RADio:STANdard:SRATe?
```

Use this command to specify symbol rate of ASK demodulation analysis.

Range: 100 sps to 100 ksps

Example: `:DMA:RADio:STANdard:SRATe 2ksps`

Key access: **Meas > Symbol Rate**

FSK Measurement Filter

```
[:SENSe]:DMA:RADio:STANdard:FILTer:MEASurement
NONE|RNYQuist|NYQuist|GAUSsian
[:SENSe]:DMA:RADio:STANdard:FILTer:MEASurement?
```

Use this command to set the measurement filter of FSK demodulation analysis.

Range: RNYQuist, NYQuist, GAUSsian, NONE

Example: `:DMA:RADio:STANdard:FILTer:MEASurement RNYQuist`

Key access: **Meas > Filter Setup > Meas Filter**

FSK Reference Filter

```
[:SENSe]:DMA:RADio:STANdard:FILTer:REFerence
RNYQuist|NYQuist|GAUSsian|NONE
[:SENSe]:DMA:RADio:STANdard:FILTer:REFerence?
```

Use this command to specify reference filter of FSK demodulation analysis.

Range: RNYQuist, NYQuist, GAUSsian, NONE

Example: `:DMA:RADio:STANdard:FILTer:REFerence RNYQuist`

Key access: **Meas > Filter Setup > Ref Filter**

FSK Filter Symbols

```
[:SENSe]:DMA:RADio:STANdard:FILTer:SYMBol <
integer >
[:SENSe]:DMA:RADio:STANdard:FILTer:SYMBol?
```

Use this command to specify FSK filter symbols.

Range: 2 to 25

Example: `:DMA:RADio:STANdard:FILTer:SYMBol 4`

Key access: **Meas > Filter Setup > FilterSymbols**

FSK Alpha/BbT

```
[:SENSe]:DMA:RADio:STANdard:ALPHa <real>
[:SENSe]:DMA:RADio:STANdard:ALPHa?
```

Use this command to specify the filter Alpha/Bbt value.

Range: 0.2 to 1

Example: `:DMA:RADio:STANdard:ALPHa 0.2`

Key access: **Meas > Filter Setup > Alpha/BbT**

FSK Measurement Length

```
[:SENSe]:DMA:MEASurement:LENGth <integer>
[:SENSe]:DMA:MEASurement:LENGth?
```

Use this command to specify measurement length of FSK demodulation analysis.

Range: 20 to 1100

Example: `:DMA:MEASurement:LENGth 200`

Key access: **Meas > Meas Length**

FSK Trigger Type

```
:TRIGger:DMA[:SEQuence]:SOURce IMMediate|EXTer-
nal|RFTRigger
:TRIGger:DMA[:SEQuence]:SOURce?
```

Use this command to specify the trigger type used in current FSK demodulation analysis.

Example: `:TRIGger:DMA:SOURce EXTernal`

Key access: **Meas > Trigger**

FSK RF Trigger Level

```
:TRIGger:DMA[:SEQuence]:LEVel:ABSolute <value>
:TRIGger:DMA[:SEQuence]:LEVel:ABSolute?
```

Use this command to turn on/off scale coupling state of X axis.

Range: –40 to 10 dBm

Example: `:TRIGger:DMA:LEVel:ABSolute 5dBm`

Key access: **Meas > Trigger > RF Trigger > Video Level**

FSK External Trigger Type

```
:TRIGger:DMA[:SEQuence]:EXTernal:SLOPe POSi-
tive|NEGative
:TRIGger:DMA[:SEQuence]:EXTernal:SLOPe?
```

This command switch external trigger type between rise slope and fall slope.

Range: POSitive, NEGative

Example: `:TRIGger:DMA:EXTernal:SLOPe NEGative`

Key access: **Meas > Trigger >External > Trigger Slope**

FSK Trigger Delay Time

```
:TRIGger:DMA[:SEQuence]:DELay <time>
:TRIGger:DMA[:SEQuence]:DELay?
```

Use this command to specify trigger delay time.

Range: 0 to 200 s

Example:
```
:TRIGger:DMA:DELay 2us
```

```
:TRIGger:DMA[:SEQuence]:DELay:STATe OFF|ON|0|1
:TRIGger:DMA[:SEQuence]:DELay:STATe?
```

Use this command to turn on/off trigger delay state.

Example:
```
:TRIGger:DMA:DELay:STATe ON
```

Key access: **Meas > Trigger > External > Trig Delay**

FSK Limit State

```
:CALCulate:DMA:LIMit OFF|ON|0|1
:CALCulate:DMA:LIMit?
```

Use this command to turn on/off limit state of FSK demodulation analysis.

Example:
```
:CALCulate:DMA:LIMit ON
```

Key access: **Shift > Limit > Limit**

FSK Carrier Power Upper Limit

```
:CALCulate:DMA:LIMit:POWer:UPPer <value>
:CALCulate:DMA:LIMit:POWer:UPPer?
```

Use this command to specify the FSK carrier power upper limit value.

Range: -100 to 30 dBm

Example:
```
:CALCulate:DMA:LIMit:POWer:UPPer 20dBm
```

Key access: **Shift > Limit > CarrPwr Upper**

FSK Frequency Deviation Upper Limit

```
:CALCulate:FSK:LIMit:FDEViation:UPPer <frequency>
:CALCulate:FSK:LIMit:FDEViation:UPPer?
```

Use this command to specify the FSK frequency deviation upper limit value.

Range: 1 Hz to 500 kHz

Example: `:CALCulate:FSK:LIMit:FDEViation:UPPer 1000`

Key access: **Shift > Limit > FreqDev Upper**

FSK Frequency Deviation Lower Limit

```
:CALCulate:FSK:LIMit:FDEViation:LOWer <frequency>
:CALCulate:FSK:LIMit:FDEViation:LOWer?
```

Use this command to specify the FSK frequency deviation lower limit value.

Range: 1 Hz to 500 kHz

Example: `:CALCulate:FSK:LIMit:FDEViation:LOWer 2000`

Key access: **Shift > Limit > FreqDev Lower**

FSK Carrier Frequency Offset Upper Limit

```
:CALCulate:DMA:LIMit:FOFFset:UPPer <frequency>
:CALCulate:DMA:LIMit:FOFFset:UPPer?
```

Use this command to specify FSK carrier freqeuncy offset upper limit.

Range: -2 MHz to 2 MHz

Example: `:CALCulate:DMA:LIMit:FOFFset:UPPer 20 kHz`

Key access: **Shift > Limit > CarrFreqOfstUp**

FSK View Type

```
:DISPlay:DMA:VIEW WAVeform|SYMBol|EYE|ERRor
:DISPlay:DMA:VIEW?
```

Use this command to toggle the different view type of ASK demodulation analyses.

Range: SYMBol, WAVeform, EYE, ERRor

Example: `:DISPlay:DMA:VIEW SYMBol`

Key access: **Shift > Disp > Display**

FSK X Scale/DIV

```
DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion <real>
DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:PDIVi-
sion?
```

Use this command to specify the scale per division of X axis in FSK demodulation analysis.

Range: 1 to 40

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:X:PDIVision 1`

Key access: **Shift > Disp > X Scale > Scale/DIV**

FSK X Reference Value

```
DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue
<integer>
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RVALue?
```

Use this command to specify the reference value of X axis in FSK demodulation analysis.

Range: -400 to 400

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:X:RVALue 2`

Key access: **Shift > Disp > X Scale > Ref Value**

FSK X Reference Position

```
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSi-
tion LEFT|CENTer|RIGHt
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:RPOSi-
tion?
```

Use this command to specify the reference position of X axis in FSK demodulation analysis.

Range: LEFT, CENTer, RIGHt

Example: `:DANalyse:ASK:VIEW:X:RPOSition CENTer`

Key access: **Shift > Disp > X Scale > Ref Position**

FSK X Scale Coupling State

```
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle
0|1|OFF|ON
:DISPlay:DMA:VIEW:WINDow:TRACe:X[:SCALe]:COUPle?
```

Use this command to turn on/off scale coupling state of X axis in FSK demodulation analysis.

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:X:COUPle ON`

Key access: **Shift > X Scale > Scale Coupling**

FSK Y Reference Value

```
:DISPlay:FSK:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue
<real>
:DISPlay:FSK:VIEW:WINDow:TRACe:Y[:SCALe]:RVALue?
```

Use this command to specify reference value of Y axis in FSK demodulation analysis.

Range: –1 to 1 MHz

Example: `:DISPlay:FSK:VIEW:WINDow:TRACe:Y:RVALue -1`

Key access: **Shift > Disp > Y Scale > Ref Value**

FSK Y Scale/DIV

```
DISPlay:FSK:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVision <real>
DISPlay:FSK:VIEW:WINDow:TRACe:Y[:SCALe]:PDIVision?
```

Use this command to specify the scale per division of Y axis.

Range: 1 Hz to 150 kHz

Example: `:DISPlay:FSK:VIEW:WINDow:TRACe:Y:PDIVision 200`

Key access: **Shift > Disp > Y Scale > Scale /DIV**

FSK Y Reference Position

```
:DISPlay:DMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSition TOP|CENTer|BOTTom
:DISPlay:DMA:VIEW:WINDow:TRACe:Y[:SCALe]:RPOSition?
```

Use this command to specify the reference position of Y axis.

Range: TOP, CENTer, BOTTom

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:Y:RPOSition TOP`

Key access: **Shift > Disp > Y Scale > Ref Position**

FSK Y Scale Coupling State

```
:DISPlay:DMA:VIEW:WINDow:TRACe:Y[:SCALe]:COUPle 0|1|OFF|ON
:DISPlay:DMA:VIEW:WINDow:TRACe:Y[:SCALe]:COUPle?
```

Use this command to turn on/off scale coupling state.

Example: `:DISPlay:DMA:VIEW:WINDow:TRACe:Y:COUPle 1`

Key access: **Shift > Disp > Y Scale > Scale Coupling**

Read FSK Trace Data

```
:TRACe[:DATA]?
```

Use this command to query FSK measurement trace data.

Example: `:TRACe?`

Power Meter Option Subsystem

The Power Meter Option Subsystem provides you the SCPI command reference for the Power Meter function. Use the command `:INSTrument POWmeter` to access this subsystem.

Option PWM or PWP is required to enable this function. Option PWM supports Agilent U2000 series USB average power sensor, and Option PWP supports Agilent U2020 X series peak power sensor.

Power Sensor Zeroing

```
:CALibration[1]:ZERO:ONCE
```

This command perform power sensor zeroing operation.

Example: `:CAL1:ZERO:ONCE`

Key access: **Meas > Zeroing > Zeroing**

Zeroing Type

```
:CALibration[1]:ZERO:TYPE EXTernal|INTernal
:CALibration[1]:ZERO:TYPE?
```

This command toggles the zeroing type between external type and internal type. U2020 X series peak power sensor only support internal zeroing type.

Example: `:CALibration1:ZERO:TYPE EXTernal`

Key access: **Meas > Zeroing > Zero Type**

Power Meter Sweep State

```
[:SENSe]:SWEep:STATe RUN|PAUSe
[:SENSe]:SWEep:STATe?
```

This command toggles power meter sweep state between run and pause.

Example: `:SWEep:STATe PAUSe`

Key access: **Meas > Pause/ Run**

Switch Power Range

```
[:SENSe]:POWer:AC:RANGe UPPer|LOWer
[:SENSe]:POWer:AC:RANGe?
```

This command toggles the measure power range between upper and lower. It's only available for U2000 series average power sensor.

Example: `:POWer:AC:RANGe UPPer`

Key access: **Meas > General Setup > Power Range > Range > Upper / Lower**

Set Power Auto Range State

```
[:SENSe]:POWer:AC:RANGe:AUTO OFF|ON|0|1
[:SENSe]:POWer:AC:RANGe:AUTO?
```

This command turns on/off power auto range state. N9322C will choose the power range according to current measurement power.

Example: `:POWer:AC:RANGe:AUTO 1`

Key access: **Meas > General Setup > Power Range > Range > Upper / Lower**

Set Power Meter Center Frequency

```
[:SENSe]:FREQuency[:CW] < freq >
[:SENSe]:FREQuency[:CW]?
```

This command defines the center frequency of power meter measurement.

Range: 1 kHz to 90 GHz

Example: `:SENSe:FREQuency:CW 100MHz`

Key access: **Meas > General Setup > Freq**

Power Meter Reference Power State

```
[:SENSe]:CORRection:CSET1:STATe OFF|ON|0|1
[:SENSe]:CORRection:CSET1:STATe?
```

Use this command to turn on/off reference power state.

Example: `:CORRection:CSET1:STATe 1`

Key access: **Meas > General Setup > Rel/Offset > Rel > On/Off**

Set Power Meter Reference Power Value

```
[:SENSe]:CORRection:CSET1[:INPut][:MAGNitude]
<value>
[:SENSe]:CORRection:CSET1[:INPut][:MAGNitude]?
```

This command defines the reference power value.

Example: –200 to +200 dBm

Example: `:CORRection:CSET1:INPut:MAGNitude 10dBm`

Key access: **Meas > General Setup > Rel/Offset > Rel**

Set Power Meter Offset State

```
[:SENSe]:CORRection:GAIN2:STATe OFF|ON|0|1
[:SENSe]:CORRection:GAIN2:STATe?
```

This command turns on/off offset state for power meter measurement.

Example: `:CORRection:GAIN2:STATe 1`

Key access: **Meas > General Setup > Rel/Offset > Offset**

Set Power Meter Offset Value

```
[:SENSe]:CORRection:GAIN2[:INPut][:MAGNitude]
<value>
[:SENSe]:CORRection:GAIN2[:INPut][:MAGNitude]?
```

This command defines the offset value of power meter measurement.

Example: `:CORRection:GAIN2:INPut:MAGNitude 15`

Key access: **Meas > General Setup > Rel/Offset > Offset**

Set Power Meter Average State

```
[:SENSe]:AVERage[:STATe] OFF|ON|0|1
[:SENSe]:AVERage[:STATe]?
```

Use this command to turn on/off power meter average state.

Example: `:AVERage:STATe 1`

Key access: **Meas > General Setup > Average > Average On/Off**

Set Power Meter Average Count Number

```
[:SENSe]:AVERage:COUNt <value>
[:SENSe]:AVERage:COUNt?
```

This command defines the power meter average count number.

Range: 1 to 1024

Example:
```
:AVERage:COUNt 10
```

Key access: **Meas > General Setup > Average > Average > Avg Count**

Set Power Meter Average Count Auto State

```
[:SENSe]:AVERage:COUNt:AUTO OFF|ON|0|1
[:SENSe]:AVERage:COUNt:AUTO?
```

This command toggles the power meter average count auto state.

Example:
```
:AVERage:COUNt:AUTO 1
```

Key access: **Meas > General Setup > Average > Average > Avg Count**

Power Meter Average Step Detect State

```
[:SENSe]:AVERage:SDETect OFF|ON|0|1
[:SENSe]:AVERage:SDETect?
```

Use this command to turn on/off step detect state.

Example:
```
:AVERage:SDETect 0
```

Key access: **Meas > General Setup > Average > Average > Step Detect On/Off**

Set Power Meter Measurement Interval Time

```
[:SENSe]:MEASure:INTerval:TIME <value>
[:SENSe]:MEASure:INTerval:TIME?
```

This command defines the power meter measurement interval time.

Range: 1 ms to 200 s

Example:
```
:MEASure:INTerval:TIME 1ms
```

Key access: **Meas > General Setup > More 1 of 2 > Meas Interval**

Set Power Meter Video Bandwidth Type

```
[:SENSe]:BANDwidth|BWIDth HIGH|MEDium|LOW|OFF
[:SENSe]:BANDwidth|BWIDth?
```

Use this command to toggle the video bandwidth type between high, middle, low and off. It's only available for U2020 X series peak power sensor.

Example: `:SENSe:BANDwidth HIGH`

Key access: **Meas > General Setup > More 1 of 2 > Video B/W**

Set Power Meter Video Average Count

```
[:SENSe]:AVERage:VIDeo:COUNt <value>
[:SENSe]:AVERage:VIDeo:COUNt?
```

This command defines the average count number of power meter. It's only available for U2020X series peak power sensor, and the trigger type must be set to continuos trigger or single trigger. The count number parameter can only be set to N-th power of 2.

```
[:SENSe]:AVERage:VIDeo:COUNt:AUTO OFF|ON|0|1
[:SENSe]:AVERage:VIDeo:COUNt:AUTO?
```

Use this command to turn on/off the average count of power meter.

Example: `:SENSe:AVERage:VIDeo:COUNt 16`

Key access: **Meas > General Setup > More 1 of 2 > Video Avg**

Set Trace X Start Time

```
:SENSe:TRACe:OFFSet:TIME <value>
:SENSe:TRACe:OFFSet:TIME?
```

This command defines the trace X start time of peak power sensor.

Range: 0 to 1 s

Example: `:SENSe:TRACe:OFFSet:TIME 0.15`

Key access: **Meas > Peak Setup > Trace Setup > X Start**

Set Trace X Scale Time

```
:SENSe:TRACe:TIME <value>
:SENSe:TRACe:TIME?
```

This command defines the trace X scale time of peak power sensor.

Range: 20 ns to 100 ms

Example: `:SENSe:TRACe:TIME 0.001`

Key access: **Meas > Peak Setup > Trace Setup > X Scale**

Set Trace Y Max Value

```
:SENSe:TRACe:YMAX:TIME <value>
:SENSe:TRACe:YMAX:TIME?
```

This command defines the trace Y max value of peak power sensor.

Range: –150 to +230 dBm

Example: `:SENSe:TRACe:YMAX:TIME 1`

Key access: **Meas > Peak Setup > Trace Setup > Y Max**

Set Trace Y Scale Value

```
:SENSe:TRACe:Y:TIME <value>
:SENSe:TRACe:Y:TIME?
```

This command defines trace Y scale value of peak power sensor.

Example: `:SENSe:TRACe:Y:TIME 1`

Key access: **Meas > Peak Setup > Trace Setup > Y Max**

Set Gate Start Time

```
[:SENSe]:SWEep[1]:OFFSet:TIME <value>
[:SENSe]:SWEep[1]|:OFFSet:TIME?
```

This command specify the gate start time of peak power meter.

Range: –1 to 1 s

Example: `:SWEep1:OFFSet:TIME 1`

Key access: **Meas > Peak Setup > Gate Setup > Start**

Set Gate Length Time

```
[:SENSe]:SWEep[1]:TIME <value>
[:SENSe]:SWEep[1]:TIME?
```

This command defines the gate length time of peak power meter measurement.

Range: 0 to 1 s

Example: `:SENSe:SWEep1:TIME 1`

Key access: **Meas > Peak Setup > Gate Setup > Length**

Set Display Resolution

```
:DISPlay[:WINDow]:NUMeric:RESolution T1|T2|T3|T4
:DISPlay[:WINDow]:NUMeric:RESolution?
```

This command toggles the readout display resolution between T1 to T4. The display resolution T4 can achieve the best resolution.

Example: `:DISPlay:WINDow:NUMeric:RESolution T4`

Key access: **Meas > Meas Disp > Resolution**

Set Display Auto Range State

```
:DISPlay[:WINDow]:DISPrange:AUTO OFF|ON|0|1
:DISPlay[:WINDow]:DISPrange:AUTO?
```

This command turns on/off the display auto range state.

Example: `:DISPlay:WINDow:DISPrange:AUTO 1`

Key access: **Meas > Meas Disp > Disp Range > Auto Range**

Set Top Value of Display Range

```
:DISPlay[:WINDow]:DISPrange:TOP <value>
:DISPlay[:WINDow]:DISPrange:TOP?
```

This command specify the top value of display range.

Range: –199.99 to +200 dBm

Example: `:DISPlay:WINDow:DISPrange:TOP 10dBm`

Key access: **Meas > Meas Disp > Disp Range > Top**

Set Bottom Value of Display Range

```
:DISPlay[:WINDow]:DISPrange:BOTTom <value>
:DISPlay[:WINDow]:DISPrange:BOTTom?
```

This command defines the bottom value of display range

Range: –200 to +199.99 dBm

Example:
```
:DISPlay:WINDow:DISPrange:BOTTom –20dBm
```

Key access: **Meas > Meas Disp > Disp Range > Bottom**

Display Mode

```
:DISPlay[:WINDow]:MODE METer|CHARt|TRACe
:DISPlay[:WINDow]:MODE?
```

This command to toggle display mode between meter, chart and trace.

Example:
```
:DISPlay:WINDow:MODE CHARt
```

Key access: **Meas > Meas Disp > Disp Mode**

Set Chart View to Home

```
:DISPlay[:WINDow]:CHARt:HOME
```

This command set the chart view page to home.

Example:
```
:DISPlay:CHARt:HOME
```

Key access: **Meas > Meas Disp > Chart View > Home**

Set Chart View to End

```
:DISPlay[:WINDow]:CHARt:END
```

This command set the chart view page to end.

Example:
```
:DISPlay:CHARt:END
```

Key access: **Meas > Meas Disp > Chart View > End**

Set Chart View to Previous

```
:DISPlay[:WINDow]:CHARt:PREVious
```

This command set the chart view page to previous page.

Example: `:DISPlay:CHARt:PREVious`

Key access: **Meas > Meas Disp > Chart View > Prev**

Set Chart View to Next

```
:DISPlay[:WINDow]:CHARt:NEXT
```

This command set the chart view page to next page.

Example: `:DISPlay:CHARt:NEXT`

Key access: **Meas > Meas Disp > Chart View > Next**

Set Limit State

```
:CALCulate1:LIMit:STATe OFF|ON|0|1
:CALCulate1:LIMit:STATe?
```

Use this command to turn on/off limit state.

Example: `:CALCulate1:LIMit:STATe 1`

Key access: **Shift > Limit > Limits**

Set Limit Upper Value

```
:CALCulate1:LIMit:UPPer[:DATA] <value>
:CALCulate1:LIMit:UPPer[:DATA]?
```

This command defines the upper limit value.

Range: –199.99 to +200 dBm

Example: `:CALCulate1:LIMit:UPPer:DATA 12`

Key access: **Shift > Limit > Upper Limit**

Set Limit Lower Value

```
:CALCulate1:LIMit:LOWer[:DATA] <value>
:CALCulate1:LIMit:LOWer[:DATA]?
```

This command defines the lower limit value.

Range: –200 to +199.99 dBm

Example:
```
:CALCulate1:LIMit:LOWer:DATA -10
```

Key access: **Shift > Limit > Lower Limit**

Set Limit Beep

```
:DISPlay[:WINDow]:DISPrange:BOTTom <value>
:DISPlay[:WINDow]:DISPrange:BOTTom?
```

Use this command to turn on/off beep warning for limit.

Example:
```
:CALCulate:LLINe:CONTrol:BEEP ON
```

Key access: **Shift > Limit > Limit Beep**

Query Limit Fail Result

```
:CALCulate:LIMit:FAIL?
```

Use this command to query the limit result. If get the result 0, it means power meter test result pass the limit. If get the result 1, it means the power meter test result fail the limit.

Example:
```
:CALCulate:LIMit:FAIL?
```

Query Limit Failure Count

```
:CALCulate:LIMit:FCOunt?
```

Use this command to query the quantity of limit fail test points.

Example:
```
:CALCulate:LIMit:FCOunt?
```

Set Gate Measurement Type

```
:CALCulate1:FEED1 PEAK|PTAV|AVER|MIN
:CALCulate1:FEED1?
```

This command to toggle gate measurement type between Average, Peak, Peak to Average, and Minimum. It's only available when the display mode is set to Meter.

Example: `:CALCulate1:FEED1 PEAK`

Key access: **Meas > Peak Setup > Gate Setup > Meas**

Set Trigger Type

```
:TRIGger1:SOURce:ACQN FREErun|CONTinuous|SINGle
:TRIGger1:SOURce:ACQN?
```

This command toggles the peak power sensor trigger type between Continuous Trigger, Single Trigger and Free Run.

Example: `:TRIGger1:SOURce:ACQN FREErun`

Key access: **Meas > Peak Setup > Trig/Acq >Acqn**

Set Trigger Source

```
:TRIGger1:SOURce EXTernal|INTernal
:TRIGger1:SOURce?
```

Use this command to toggle the trigger source between external and internal. It's only available for continuous trigger or single trigger.

Example: `:TRIGger1:SOURce EXTernal`

Key access: **Meas > Peak Setup > Trig/Acq > Source**

Power Meter Auto Trigger Level State

```
:TRIGger[:SEQuence]:LEVel:AUTO OFF|ON|0|1
:TRIGger[:SEQuence]:LEVel:AUTO?
```

This command turns on/off auto trigger level state of power meter measurement.

Example: `:TRIGger:SEQuence:LEVel:AUTO 1`

Key access: **Meas > Peak Setup > Trig/Acq > Mode**

Set Trigger Level

```
:TRIGger[:SEQuence]:LEVel <value>
```

```
:TRIGger[:SEQuence]:LEVel?
```

This command defines the trigger level of power sensor. It's only available for continuous trigger or single trigger. The trigger source must be internal, and auto trigger level state must be off (set to normal mode).

Range: –100 to 0 dBm

Example: `:TRIGger:SEQuence:LEVel -20dBm`

Key access: **Meas > Peak Setup > Trig/Acq > Level**

Power Meter Trigger Delay Time

```
:TRIGger[:SEQuence]:DELay <value>
```

```
:TRIGger[:SEQuence]:DELay?
```

This command defines trigger delay time of power meter measurement.

Range: 0 to 1 s

Example: `:TRIGger:SEQuence:DELay 1`

Key access: **Meas > Peak Setup > Trig/Acq > Delay**

Power Meter Trigger Slope Mode

```
:TRIGger[:SEQuence]:SLOPe POSitive|NEGative
```

:TRIGger[:SEQuence]:SLOPe?

This command toggles trigger slope mode between rise and fall.

Example: `:TRIGger:SEQuence:SLOPe POSitive`

Key access: **Meas > Peak Setup > Trig/Acq > More 1 of 2 > Slop**

Power Meter Trigger Holdoff Time

```
TRIGger[:SEQuence]:HOLDoff <value>
TRIGger[:SEQuence]:HOLDoff?
```

This command defines the trigger hold off time of power meter.

Range: 1 us to 400 ms

Example: `:TRIGger:SEQuence:HOLDoff 0.1`

Key access: **Meas > Peak Setup > Trig/Acq > More 1 of 2 > Holdoff**

Power Meter Trigger Hysteresis Value

```
:TRIGger[:SEQuence]:HYSTeresis <value>
:TRIGger[:SEQuence]:HYSTeresis?
```

This command defines the trigger hysteresis value of power meter.

Example: `:TRIGger:SEQuence:HYSTeresis 1`

Key access: **Meas > Peak Setup > Trig/Acq > More 1 of 2 > Hysteresis**

Tracking Generator Option Subsystem

The Tracking Generator Option Subsystem provides you the SCPI command reference for the Tracking Generator funciton.Use the command `:INSTrument TGENerator` to access this subsystem.

Option TG7 is required to enable this function.

Tracking Generator Output Amplitude

```
:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] <value>
:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]?
```

***RST:** –20 dBm

Range: 0 to –30 dBm

Example: `:SOURce:POWer -10 dBm`

Key access: **Meas > Amplitude**

Tracking Generator Amplitude On/Off

```
:OUTPut[:STATe] OFF|ON|0|1
:OUTPut[:STATe]?
```

Use this command to turn/off the tracking generator output.

***RST:** Off

Key access: **Meas > Amplitude On Off**

Tracking Generator Amplitude Offset

```
:SOURce:CORRection:OFFSet <value>
:SOURce:CORRection:OFFSet?
```

This command sets the amplitude offset of the tracking generator output. This command is valid after tracking generator is enabled.

***RST:** 0 dB

Range: 0 to –30 dBm

Example: `:SOURce:CORRection:OFFSet -10 dB`

Key access: **Meas > Amptd Offset**

Tracking Generator Amplitude Step

```
:SOURce:POWer:STEP[:INCRement] <value>
:SOURce:POWer:STEP[:INCRement]?
```

This command sets the amplitude step size of the tracking generator output. This command is valid after tracking generator is enabled.

Range: 0 to –10 dBm

Example: `:SOURce:POWer:STEP 1 dB`

Key access: **Meas > Amptd Step**

Tracking Generator Storing as Reference

```
:CALCulate:NTData:STORe
```

This command stores the current trace to Trace 4 as a reference. This command is valid after tracking generator is enabled.

Key access: **Meas > Normalize >Store Ref**

Tracking Generator Normalization State

```
:CALCulate:NTData[:STATe] OFF|ON|0|1
:CALCulate:NTData[:STATe]?
```

Use this command to turn on/off tracking generator normalization.

***RST:** Off

Example: `:CALCulate:NTData 1`

Key access: **Meas > Normalize > Normalize On/Off**

Normalization Reference Level

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel <value>
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel?
```

Use this command to set the normalization reference level.

***RST:** 0 dB

Range: –327.6 to 327.6 dB

Example: `:DISPlay:WINDow:TRACe:Y:NRLevel -10 dB`

Key access: **Meas > Normalize > Norm Ref Lvl**

Normalization Reference Position

```
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition <integer>
:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition?
```

Use this command to set the normalization reference position.

***RST:** 10

Range: 0 to 10

Example: `:DISPlay:WINDow:TRACe:Y:NRPosition 10`

Key access: **Meas > Normalize > Norm Ref Lvl**