Programmer Manual

Tektronix

RSA2203A & RSA2208A
3 GHz & 8 GHz Real-Time Spectrum Analyzers

071-1336-04

This document applies to firmware version 3.10
and above.

www.tektronix.com

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

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

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

Contacting Tektronix

Tektronix, Inc.
14200 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

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

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

Table of Contents

Getting Started

Syntax and Commands

Status and Events

Programming Examples

Appendices

Glossary and Index

List of Figures

List of Tables

Preface

This programmer manual is for the RSA2203A and RSA2208A Real-Time Spectrum Analyzers. It provides information on operating your analyzer using the General Purpose Interface Bus (GPIB).

This manual is composed of the following sections:

- *Getting Started* outlines how to use the GPIB interface.
- *Syntax and Commands* defines the syntax used in command descriptions, presents a list of all command subsystems, and presents detailed descriptions of all programming commands.
- *Status and Events* describes how the status and Events Reporting system operates and presents a list of all system errors.
- *Programming Examples* describes some example analyzer programs.
- *Appendices* provides additional information including character charts, GPIB interface specification, and factory initialization settings.

Related Manual

RSA2203A and RSA2208A User Manual
(Standard accessory; Tektronix part number 071-1334-XX)
Describes how to install the analyzer and how to work with the menus and details the functions.

Difference between RSA2203A and RSA2208A

RSA2203A and RSA2208A have the same functions except for their measurement frequency ranges:

RSA2203A . . . . . DC to 3 GHz
RSA2208A . . . . . DC to 8 GHz

Unless otherwise noted, descriptions in this manual apply to both.

Getting Started

Getting Started

You can write computer programs that remotely set the analyzer front panel controls or that take measurements and read those measurements for further analysis or storage.

To help you get started with programming the analyzer, this section includes the following sections:

- *Overview of the Manual*
  Summarizes the type of programming information contained in each major section of this manual.
- *Connecting the Interface*
  Describes how to physically connect the analyzer to a controller.
- *Using GPIB Ports*
  Describes how to use the GPIB port.
- *Setting the GPIB Parameters from the Front Panel*
  Describes how to set the GPIB parameters from the front panel.
- *Using TekVISA*
  Describes how to use the TekVISA communication protocol.

Overview of the Manual

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

Syntax and Commands

Section 2, *Syntax and Commands*, describes the structure and content of the messages your program sends to the analyzer. Figure 1–1 shows command parts as described in the *Command Syntax* subsection.

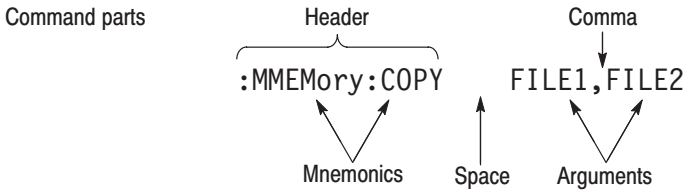


Figure 1–1: Command parts

Section 2 also describes the effect of each command and provides examples of how you might use it. The *Command Groups* section provides lists by functional areas. The *IEEE Common Commands* and the subsequent sections arrange commands alphabetically (Figure 1–2).

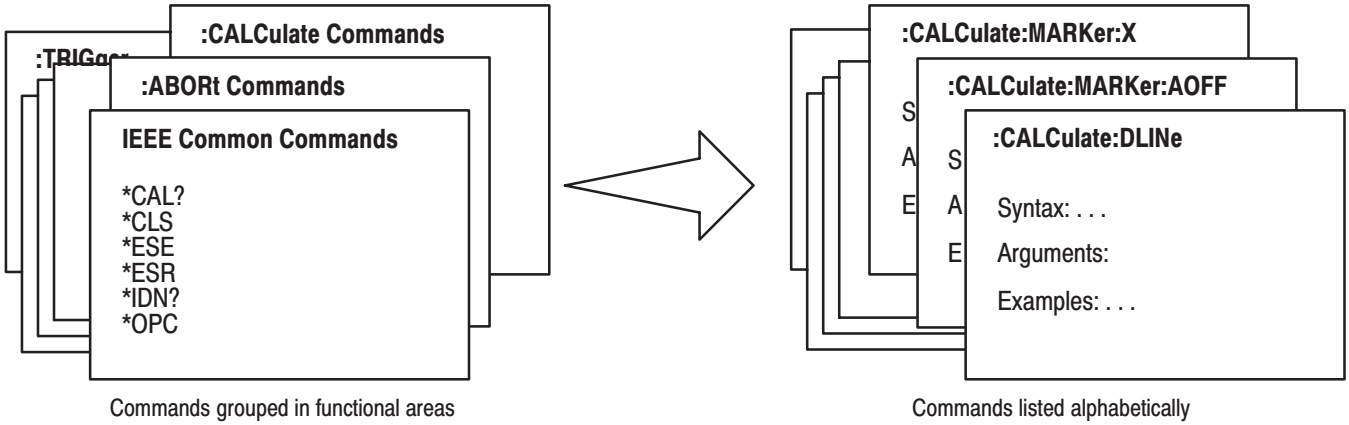


Figure 1–2: Functional groupings and an alphabetical list of commands

Status and Events

The program may request information from the analyzer. The analyzer provides information in the form of status and error messages. Figure 1–3 illustrates the basic operation of this system.

Section 3, *Status and Events*, describes how to get status or event information from the program and details the event and error messages.

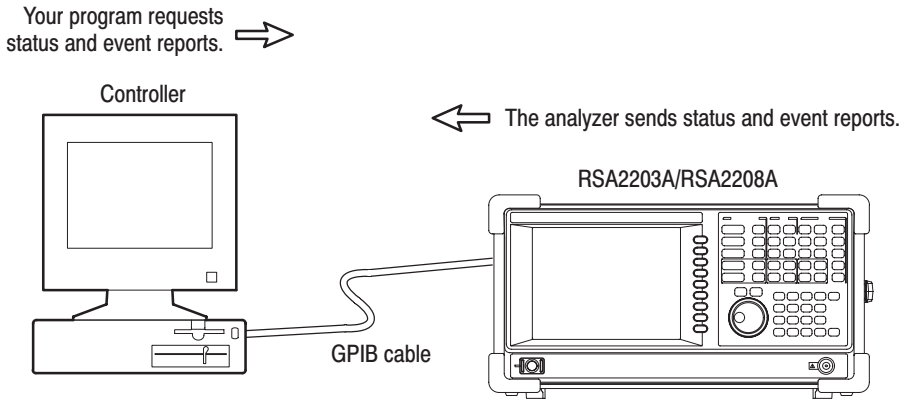


Figure 1–3: Event-driven program

Programming Examples

Section 4, *Programming Examples*, includes Visual C++ source code as well as sample programs for running macro programs.

```
GpibWrite("INSTrument 'SANORMAL'");
GpibWrite("*RST");
GpibTimeOut(NORMAL_TIME);
GpibWrite("CONFigure:SPECtrum:CHPower");
GpibWrite("FREQuency:BAND RF1B");
GpibWrite("FREQuency:CENTer 1GHz");
GpibWrite("FREQuency:SPAN 1MHz");
GpibWrite("*CAL?");
GpibRead(readBuf, MAX_BUF);
printf("*CAL? result = %s\n", readBuf);
GpibWrite("CHPower:BANDwidth:INTegration 300kHz");
GpibWrite("SPECtrum:AVERage ON");
```

Figure 1–4: Sample program (Visual C++ source code)

Connecting the Interface

The analyzer has a 24-pin GPIB connector on its rear panel, as shown in Figure 1–5. This connector has a D-type shell and conforms to IEEE Std 488.1–1987.

Attach an IEEE Std 488.1-1987 GPIB cable (Tektronix part number 012-0991-00) to this connector.

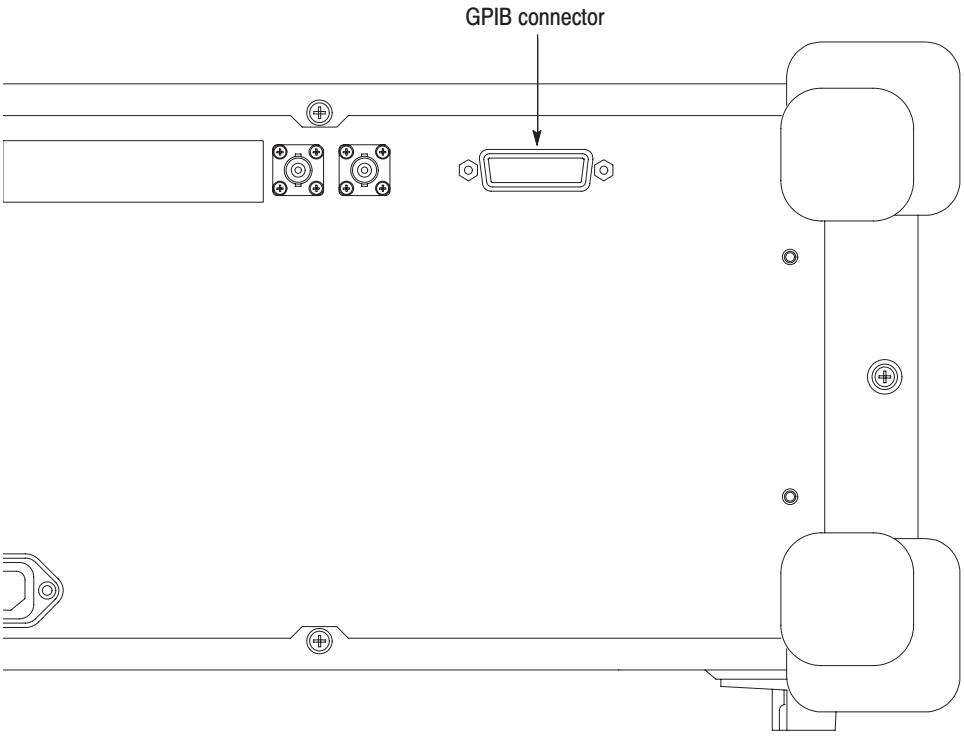


Figure 1–5: GPIB connector (rear panel)

Appendix B: GPIB Interface Specifications gives more information on the GPIB configuration of the analyzer.

For the other interfaces, refer to the *RSA2203A and RSA2208A User Manual*.

Using the GPIB Port

The analyzer has Talker/Listener functions through which it can communicate with other devices, as well as the external controller, located on the bus.

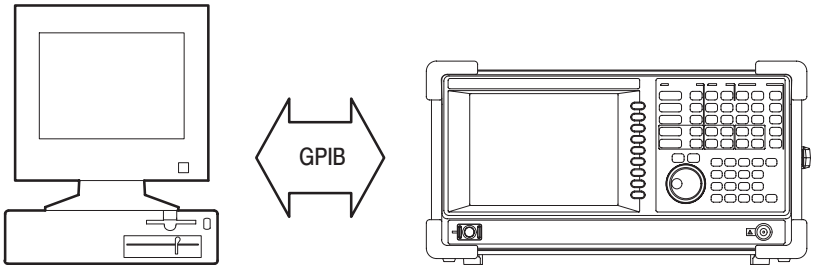


Figure 1–6: GPIB connection

GPIB Requirements

Observe these rules when you use your analyzer with a GPIB network:

- Assign a unique device address to each device on the bus. No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 meters (6 feet) of cable used.
- Do not use more than 20 meters (65 feet) of cable to connect devices to a bus.
- Turn on at least two-thirds of the devices on the network while using the network.
- Connect the devices on the network in a star or linear configuration as shown in Figure 1–7. Do not use loop or parallel configurations.

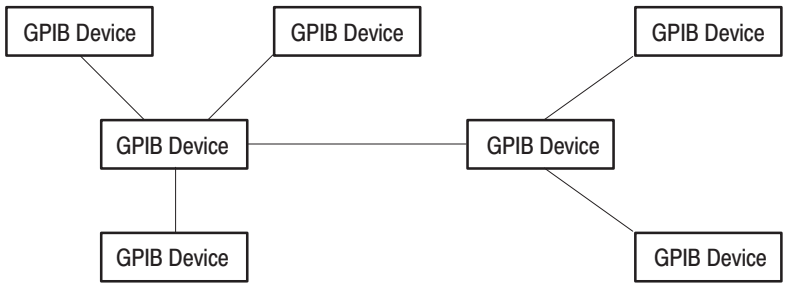


Figure 1–7: Typical GPIB network configurations

Setting the GPIB Parameters from the Front Panel

Use the **SYSTEM → Remote Setup** menu to set the GPIB parameters as required for the bus configuration. Once you have set the parameters, you can control the analyzer through the GPIB interface.

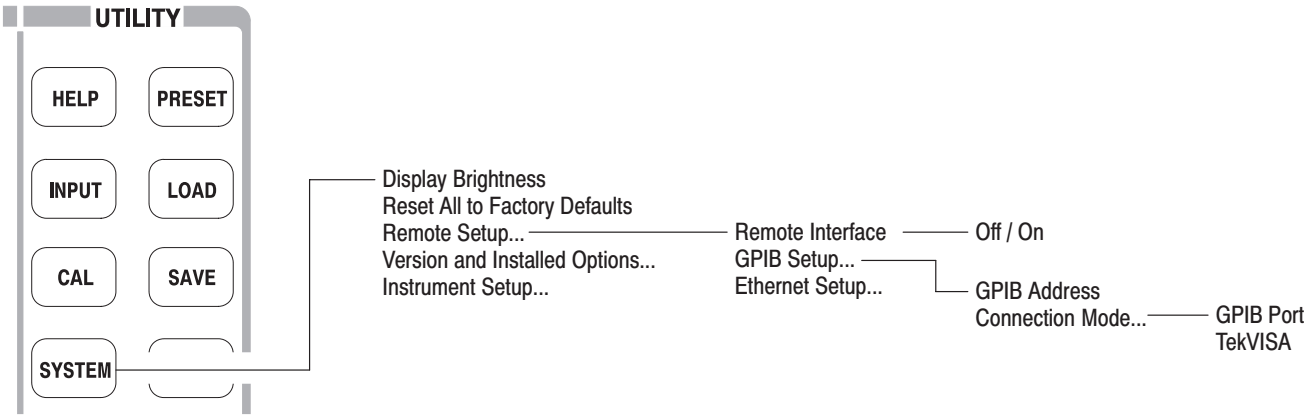


Figure 1–8: Remote Setup menu

Remote Setup Menu

The Remote Setup menu contains the following controls:

Remote Interface. Turns on or off the connection between the analyzer and the interface bus.

GPIB Setup... Sets the GPIB address and connection mode.

GPIB Address. Sets the GPIB address of the analyzer when GPIB Port is selected as the Connection Mode. Range: 0 to 30 (default: 1)

Connection Mode. Selects the physical GPIB port or the virtual (TekVISA) connection method.

- **GPIB Port.** *Default.* Uses the IEEE488.2 connector on the rear panel of the analyzer to communicate with an external controller.
  Refer to the next section *Setting up the GPIB port* for the procedure.
- **TekVISA.** Uses TekVISA to communicate with test instrumentation through Ethernet (LAN connector on the side panel of the analyzer), and also to run a control program locally on the analyzer.
  Refer to *Using TekVISA* on page 1–8 for more information.

Ethernet Setup... Not available currently. Use the Windows XP Control Panel to set up networking parameters.

Setting Up the GPIB Port

When you use the GPIB port, follow these steps to set the parameters:

1. Press the **SYSTEM** key in the UTILITY block on the front panel.
2. Press the side key **Remote Setup...→ GPIB Setup...**

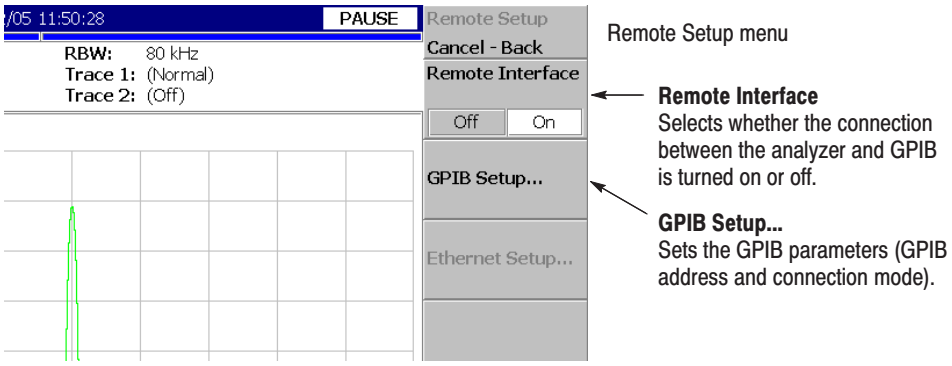


Figure 1–9: Setting the GPIB parameters

3. Press the **Connection Mode...** side key and select **GPIB Port**.
4. Press the **GPIB Address** side key and set the address using either the general purpose knob or the numeric keypad.

***NOTE**. The GPIB address cannot be initialized with* *RST *command.*

5. Press the **Cancel–Back** (top) side key and then the **Remote Interface** side key to select **On**.

To disconnect the analyzer from the bus:

- Press the **Remote Interface** side key to select **Off**.

When the analyzer is disconnected from the bus, all the communication processes with the controller are interrupted.

Using TekVISA

TekVISA is Tektronix implementation of VISA (Virtual Instrument Software Architecture), an industry-standard communication protocol. It allows you to write programs using the WCA200A Series SCPI command set to control the instrument through interfaces besides the built-in IEEE 488.2 port. Programs are written to execute on the local or remote controller. The WCA200A Series implementation of TekVISA includes a subset of the TekVISA functionality offered on Tektronix oscilloscopes. The Virtual GPIB (GPIB8), GPIB, and LAN (VXI-11 protocol) interfaces are supported, but not the ASRL interface.

***NOTE**. The details on TekVISA concepts and operations are explained in the* TekVISA Programmer Manual. *Refer to* Installing TekVISA *described below for accessing the files.*

Be aware of the following points:

- If TekVISA is not installed or has not been activated, and you select TekVISA as the connection mode, the instrument still attempts to connect to TekVISA. This does not hang up the instrument, but the GPIB port is taken off-line until you select GPIB Port as the connection mode again.
- Applications which are designed to execute locally on the instrument need to share the Windows processor with the measurement calculation software of the analyzer. If the controller application is very compute-intensive, it will slow down the analyzer application significantly.

Installing TekVISA

The TekVISA tools are not installed when you receive the instrument. Use the following procedure to install the tools.

To use TekVISA, these conditions must be satisfied:

- Windows XP is used as the instrument's operating system. Instruments using Windows 98SE must be upgraded to Windows XP for TekVISA to operate properly.
- A TekVISA-compatible version of the analyzer application is installed and running on the instrument. Version must be greater than 3.00.000.
- TekVISA is installed on the instrument. Version 2.03 is recommended.

The TekVISA-related files are on the internal hard disk of the analyzer in these directories:

- *C:\Tektronix\TekVISA\installer* contains the TekVISA installer.
- *C:\Tektronix\TekVISA\manual* contains the TekVISA Programmer Manual.

Use the following steps to install the TekVISA tools on your analyzer:

***NOTE**. For details on accessing Windows XP on the analyzer, refer to the* RSA2203A and RSA2208A User Manual.

1. Connect a USB mouse and keyboard to the USB ports on the side panel of the analyzer.
2. Display the Windows XP desktop on the screen.
3. Find the *setup.exe* file in the *C:\Tektronix\TekVISA\installer* directory using Windows Explorer or other file access methods.
4. Run *setup.exe* and follow the instructions.

The *TekVISA Programmer Manual* is found in the *C:\Tektronix\TekVISA\manual* directory.

Syntax and Commands

Command Syntax

This section contains information on the Standard Commands for Programmable Instruments (SCPI) and IEEE 488.2 Common Commands you can use to program your RSA2203A/RSA2208A analyzer. The information is organized in the following subsections:

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

Backus-Naur Form Definition

This manual may describe commands and queries using the Backus-Naur Form (BNF) notation. Table 2–1 defines the standard BNF symbols:

Table 2–1: BNF symbols and meanings

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

SCPI Commands and Queries

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

The SCPI language is based on a hierarchical or tree structure (see Figure 2–1) that represents a subsystem. The top level of the tree is the root node; it is followed by one or more lower-level nodes.

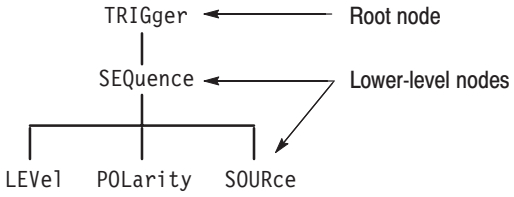


Figure 2–1: Example of SCPI subsystem hierarchy tree

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

Creating Commands

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

In Figure 2–1, TRIGger is the root node and SEQuence, LEVel, POLarity, and SOURce are lower-level nodes. To create a SCPI command, start with the root node TRIGger and move down the tree structure adding nodes until you reach the end of a branch. Most commands and some queries have parameters; you must include a value for these parameters. If you specify a parameter value that is out of range, the parameter will be set to a default value. The command descriptions, which start on page 2–33, list the valid values for all parameters.

For example, TRIGger:SEQuence:SOURce EXT is a valid SCPI command created from the hierarchy tree in Figure 2–1.

Creating Queries

To create a query, start at the root node of a tree structure, move down to the end of a branch, and add a question mark. TRIGger:SEQuence:SOURce? is an example of a valid SCPI query using the hierarchy tree in Figure 2–1.

Query Responses

The query causes the analyzer to return information about its status or settings. When a query is sent to the analyzer, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format, as shown in Table 2–2.

Table 2–2: Query response examples

Query	Response
:DISPlay:OVIew:SGRam:X:SPAN?	10.0E+6
:SENSe:AVERage:TYPE?	RMS

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

Parameter Types

Every parameter in the command and query descriptions is of a specified type. The parameters are enclosed in brackets, such as <value>. The parameter type is listed after the parameter and is enclosed in parentheses, for example, (discrete). Some parameter types are defined specifically for the RSA2200 Series command set and some are defined by ANSI/IEEE 488.2-1987 (refer to Table 2–3).

Table 2–3: Parameter types used in syntax descriptions

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

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

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

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

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

SCPI-defined Parameters. In addition to the ANSI/IEEE 488.2-1987-defined parameters, RSA2200 Series support the following SCPI-defined parameters.

- <NRf> for boolean

  ```
  OFF | ON | 0 | 1 | <NRf>
  ```

  You can use <NRf> for boolean parameter. The values other than zero (OFF) are regarded as one (ON).

- MAXimum and MINimum for numeric parameters

  You can use MAXimum and MINimum for the numeric parameter <NRf>. The following example sets the trigger level to the maximum (100%).

  ```
  :TRIGger[:SEQuence]:LEVel:IF MAXimum
  ```

  The commands that have numeric parameters support the following query:

  ```
  <header>? { MAXimum | MINimum }
  ```

  The query command returns the maximum or minimum acceptable value for the command. For example,

  ```
  :TRIGger[:SEQuence]:LEVel:IF? MAXimum
  ```

  returns 100 indicating the maximum trigger level is 100%.

- UP and DOWN for numeric parameters

  The [:SENse]:FREQuency:CENTer command (refer to page 2–272) supports UP and DOWN for the numeric parameters. The increment/decrement of UP/DOWN is determined by one of these commands:

  ```
  [:SENse]:FREQuency:CENTer:STEP:AUTO
  [:SENse]:FREQuency:CENTer:STEP[:INCRement]
  ```

Special Characters

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

Abbreviating Commands, Queries, and Parameters

You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these short forms as a combination of upper and lower case letters. The upper case letters indicate the accepted short form of a command. As shown in Figure 2–2, you can create a short form by using only the upper case letters. The accepted short form and the long form are equivalent and request the same action of the instrument.

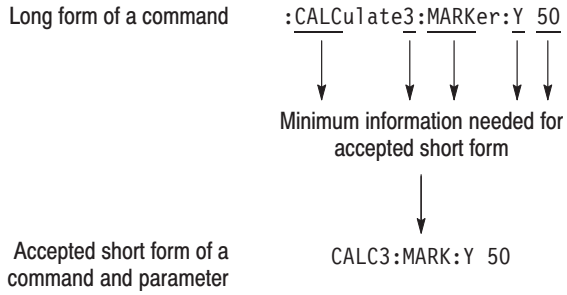


Figure 2–2: Example of abbreviating a command

NOTE*. The numeric suffix of a command or query may be included in either the long form or short form; the analyzer will default to "1" if no suffix is used. In Figure 2–2, the "3" of "CALC3" indicates that the command is directed to View 3.*

Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, add a semicolon (;), and then add more commands or queries and semicolons until the message is complete. If the command following a semicolon is a root node, precede it with a colon (:). Figure 2–3 illustrates a chained message consisting of several commands and queries. The single chained message should end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.

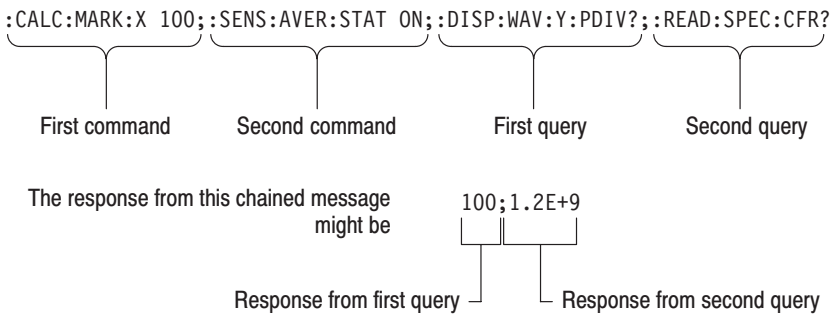


Figure 2–3: Example of chaining commands and queries

If a command or query has the same root and lower-level nodes as the previous command or query, you can omit these nodes. In Figure 2–4, the second command has the same root node (TRIG:SEQuence) as the first command, so these nodes can be omitted.

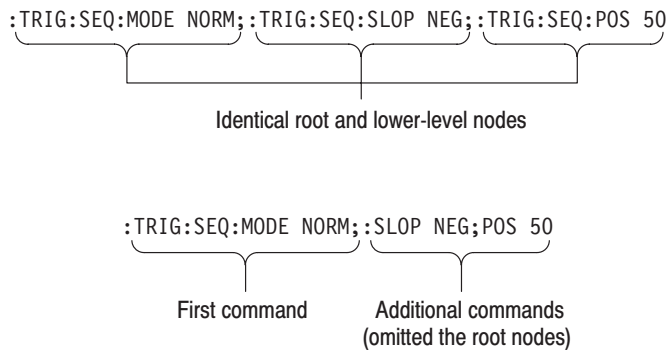


Figure 2–4: Example of omitting root and lower-level nodes in a chained message

Unit and SI Prefix

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

Table 2–4 lists the available units:

Table 2–4: Available units

Symbol	Meaning
dB	decibel (relative amplitude)
dBm	decibel (absolute amplitude)
DEG	degree (phase)
Hz	hertz (frequency)
PCT	percent (%)
s	second (time)
V	volt

The available SI prefixes are shown in Table 2–5 below:

Table 2–5: Available SI prefixes

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

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

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

15.0E6, 1.5E7Hz, 15000000, 15000000Hz, 15MHz, etc.
("15M" is not allowed.)

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

170mhz, 170mHz, 170MHz, etc.
250mv, 250mV, 250MV, etc.

General Rules

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

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

  correct: “This string uses quotation marks correctly.”

  correct: ‘This string also uses quotation marks correctly.’

  incorrect: “This string does not use quotation marks correctly.’

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

  ```
  SENSE:SPECTRUM:FFT:LENGTH 1024
  ```

  is the same as

  ```
  sense:spectrum:fft:length 1024
  ```

  and

  ```
  SENSE:spectrum:FFT:length 1024
  ```

***NOTE**. Literal strings (quoted) are case sensitive. For example: file names.*

- No embedded spaces are allowed between or within nodes.

  correct: `SENSE:SPECTRUM:FFT:LENGTH 1024`

  incorrect: `SENSE: SPECTRUM: FFT: LEN GTH 1024`

IEEE 488.2 Common Commands

Description

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

Command and Query Structure

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

- `*ESE 16`
- `*CLS`

The following are examples of common queries:

- `*ESR?`
- `*IDN?`

Constructed Mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic can be either CALCulate1, CALCulate2, CALCulate3, or CALCulate4. You use these mnemonics in the command just as you do any other mnemonic. For example, there is a :CALCulate1:MARKer:MODE command, and there is also a :CALCulate2:MARKer:MODE command. In the command descriptions, this list of choices is abbreviated as CALCulate<x>. The value of <x> is the upper range of valid suffixes. If the numeric suffix is omitted, the analyzer uses the default value of "1".

Table 2–6: Constructed mnemonics

Symbol	Meaning
CALCulate<x>	A view specifier where <x> = 1 to 4.
DLINe<x>	A horizontal display line specifier where <x> = 1 or 2.
VLINe<x>	A vertical display line specifier where <x> = 1 or 2.
MARKer<x>	A marker specifier where <x> = 1 or 2.
TRACe<x> DATA<x>	A trace specifier where <x> = 1 or 2.

Command Groups

This section lists the RSA2200 Series analyzer commands in two ways. It first presents them by functional groups. It then lists them alphabetically. The functional group list starts below. The alphabetical list provides more detail on each command and starts on page 2–33.

The RSA2200 Series analyzers conform to the Standard Commands for Programmable Instruments (SCPI) 1999.0 and IEEE Std 488.2-1987 except where noted.

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

Each command may be available or unavailable, depending on the current measurement mode. The "Measurement Modes" item in each command description shows the measurement mode in which the command is available. To set the measurement mode, use the :INSTrument[:SELect] command (refer to page 2–191) using one of the mnemonics listed below:

Table 2–7: Measurement mode

Mnemonic	Meaning
SANORMAL	Normal spectrum analysis
SASGRAM	Spectrum analysis with spectrogram
SARTIME	Real-time spectrum analysis
SAZRTIME	Real-time spectrum analysis with zoom function
DEMADEM	Analog modulation analysis
TIMCCDF	CCDF analysis
TIMTRAN	Time characteristic analysis
TIMPULSE	Pulse characteristics analysis

For the conventions of notation in this manual, refer to *Command Syntax* on page 2-1 and following pages.

Functional Groups

The commands are divided into the groups listed below.

Table 2–8: List of command groups

Command group	Function
IEEE common	Conforms to the IEEE Std 488.2-1987.
:ABORt	Resets and restarts sweep, measurement, and trigger.
:CALCulate	Controls the markers and the display line.
:CALibration	Calibrates the analyzer.
:CONFigure	Configures the analyzer for each measurement session.
:DISPlay	Controls how to show waveform and measurement result on screen.
:FETCh	Retrieves the measurements from the data last acquired.
:FORMat	Sets the output data format.
:HCOPy	Controls screen hardcopy.
:INITiate	Controls data acquisition.
:INPut	Sets the input-related conditions.
:INSTrument	Selects a measurement mode.
:MMEMory	Controls file saving/loading to/from the hard disk or floppy disk.
:PROGram	Controls macro programs.
:READ	Obtains the measurement results with acquiring data.
:SENSe	Sets up detailed conditions for each measurement.
:STATus	Controls the status and event registers.
:SYSTem	Sets system parameters and queries system information.
:TRACe	Controls display of Trace 1 and 2.
:TRIGger	Controls triggering.
:UNIT	Specifies fundamental units for measurement.

The following sections list the commands by group.

IEEE Common Commands

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

Table 2–9: IEEE common commands

Header	Description
*CAL?	Runs all the calibration routines.
*CLS	Clears the status or event.
*ESE(?)	Sets the value for the ESER register.
*ESR?	Queries the SESR register value.
*IDN?	Queries the analyzer ID.
*OPC(?)	Synchronizes commands.
*OPT?	Queries the options incorporated in the analyzer.
*RST	Restores the factory initialization settings.
*SRE(?)	Sets the value for the SRER register.
*STB?	Queries the Status Byte Register value.
*TRG	Generates a trigger event.
*TST?	Runs a self test.
*WAI	Waits until the run of another command is completed.

:ABORt Commands

Resets the trigger system and related actions such as data acquisition and measurement.

Table 2–10: :ABORt commands

Header	Description
:ABORt	Resets and restarts sweep, trigger, and measurement.

:CALCulate Commands

Control the marker and the display line.

Table 2–11: :CALCulate commands

Header	Description
:CALCulate<x>:DLINe<y>(?)	Sets the vertical position of the horizontal line.
:CALCulate<x>:DLINe<y>:STATe(?)	Determines whether to show the horizontal line.
:CALCulate<x>:MARKer:AOFF	Turns off all the markers.
:CALCulate<x>:MARKer<y>:MAXimum	Places the marker at the maximum point on the trace.
:CALCulate<x>:MARKer<y>:MODE(?)	Selects the marker mode (position or delta).
:CALCulate<x>:MARKer<y>:PEAK:HIGHer	Moves the marker to the next higher peak.
:CALCulate<x>:MARKer<y>:PEAK:LEFT	Moves the marker to the peak on the left.
:CALCulate<x>:MARKer<y>:PEAK:LOWer	Moves the marker to the next lower peak.
:CALCulate<x>:MARKer<y>:PEAK:RIGHt	Moves the marker to the peak on the right.
:CALCulate<x>:MARKer<y>:PTHReshold(?)	Sets the minimum jump of the marker on the horizontal axis.
:CALCulate<x>:MARKer<y>:RCURsor	Displays the reference cursor at the marker position.
:CALCulate<x>:MARKer<y>:ROFF	Turn off the reference cursor.
:CALCulate<x>:MARKer<y>[:SET]:CENTer	Sets the center frequency to the value at the marker position.
:CALCulate<x>:MARKer<y>[:SET]:MEASurement	Sets the measurement position with the marker.
:CALCulate<x>:MARKer<y>[:STATe]	Determines whether to show the marker.
:CALCulate<x>:MARKer<y>:TOGGle	Replaces the delta marker with the main marker.
:CALCulate<x>:MARKer<y>:TRACe(?)	Selects the trace to place the marker.
:CALCulate<x>:MARKer<y>:X(?)	Positions the marker on the horizontal axis.
:CALCulate<x>:MARKer<y>:Y(?)	Positions the marker on the vertical axis.
:CALCulate<x>:VLINe<y>(?)	Sets the horizontal position of the vertical line.
:CALCulate<x>:VLINe<y>:STATe(?)	Determines whether to show the vertical line.

:CALibration Commands

Calibrate the analyzer.

Table 2–12: :CALibration commands

Header	Description
:CALibration[:ALL](?)	Runs all the calibration routines.
:CALibration:AUTO(?)	Determines whether to run the RF gain calibration automatically.
:CALibration:DATA:DEFault	Restores the calibrated data to the factory defaults.
:CALibration:OFFSet:BASebanddc(?) (Option 05 only)	Runs the baseband DC offset calibration.
:CALibration:OFFSet:CENTer(?)	Runs the center offset calibration.
:CALibration:RF(?)	Runs the RF gain calibration.

:CONFigure Commands

Set up the analyzer in order to perform the specified measurement.

Table 2–13: :CONFigure commands

Header	Description
:CONFigure:ADEMod:AM	Sets up the analyzer to the AM signal analysis default settings.
:CONFigure:ADEMod:FM	Sets up the analyzer to the FM signal analysis default settings.
:CONFigure:ADEMod:PM	Sets up the analyzer to the PM signal analysis default settings.
:CONFigure:ADEMod:PSPectrum	Sets the analyzer to the pulse spectrum measurement default settings.
:CONFigure:CCDF	Sets the up analyzer to the CCDF measurement default settings.
:CONFigure:OVIew	Turns off measurement to obtain display data in the overview
:CONFigure:PULSe	Sets the analyzer to the pulse characteristics measurement default settings.
:CONFigure:SPECtrum	Sets up the analyzer to the spectrum measurement default settings.
:CONFigure:SPECtrum:ACPower	Sets up the analyzer to the ACPR measurement default settings.
:CONFigure:SPECtrum:CFRequency	Sets up the analyzer to the carrier frequency measurement default settings.
:CONFigure:SPECtrum:CHPower	Sets up the analyzer to the channel power measurement default settings.
:CONFigure:SPECtrum:CNRatio	Sets up the analyzer to the C/N measurement default settings.
:CONFigure:SPECtrum:EBWidth	Sets up the analyzer to the emission bandwidth measurement default settings.
:CONFigure:SPECtrum:OBWidth	Sets up the analyzer to the OBW measurement default settings.
:CONFigure:SPECtrum:SPURious	Sets up the analyzer to the spurious signal measurement default settings.
:CONFigure:TFRequency:RTIMe	Sets up the analyzer to the real-time spectrum measurement default settings.
:CONFigure:TFRequency:SGRam	Sets up the analyzer to the spectrogram measurement default settings.

Table 2–13: :CONFigure commands (Cont.)

Header	Description
:CONFigure:TRANsient:FVTime	Sets up the analyzer to the frequency vs. time measurement default settings.
:CONFigure:TRANsient:IQVTime	Sets up the analyzer to the IQ level vs. time measurement default settings.
:CONFigure:TRANsient:PVTime	Sets up the analyzer to the power vs. time measurement default settings.

:DISPlay Commands

Control how to show measurement data on the screen.

Table 2–14: :DISPlay commands

Header	Description
:DISPlay:CCDF subgroup	CCDF measurement related.
:DISPlay:CCDF:LINE:GAUSsian[:STATe](?)	Determines whether to show the Gaussian line.
:DISPlay:CCDF:LINE:REFerence[:STATe](?)	Determines whether to show the reference line.
:DISPlay:CCDF:LINE:REFerence:STORe	Stores the current CCDF trace as the reference line.
:DISPlay:CCDF:X[:SCALe]:AUTO(?)	Determines whether to set the horizontal scale automatically.
:DISPlay:CCDF:X[:SCALe]:MAXimum(?)	Sets the maximum horizontal value (right end).
:DISPlay:CCDF:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value (left end).
:DISPlay:CCDF:Y[:SCALe]:FIT	Runs auto-scale.
:DISPlay:CCDF:Y[:SCALe]:FULL	Sets the vertical axis to the default full-scale.
:DISPlay:CCDF:Y[:SCALe]:MAXimum(?)	Sets the maximum vertical value (top end).
:DISPlay:CCDF:Y[:SCALe]:MINimum(?)	Sets the minimum vertical value (bottom end).
:DISPlay:OVIew subgroup	DEMOD and TIME mode overview related.
:DISPlay:OVIew:FORMat(?)	Selects the overview display format.
:DISPlay:OVIew:SGRam:COLor[:SCALe]:OFFSet(?)	Sets the minimum color-axis value (bottom end) of the spectrogram.
:DISPlay:OVIew:SGRam:COLor[:SCALe]:RANGe(?)	Sets the color-axis full-scale of the spectrogram.
:DISPlay:OVIew:SGRam:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value (left end) of the spectrogram.
:DISPlay:OVIew:SGRam:X[:SCALe]:SPAN(?)	Sets the horizontal full-scale (span) of the spectrogram.
:DISPlay:OVIew:SGRam:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical value of the spectrogram (bottom end).
:DISPlay:OVIew:SGRam:Y[:SCALe]:PLINe(?)	Sets the vertical scale of the spectrogram.
:DISPlay:OVIew:WAVeform:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value (left edge) in the time domain display.
:DISPlay:OVIew:WAVeform:X[:SCALe]:PDIVision(?)	Sets the horizontal scale in the time domain display.
:DISPlay:OVIew:WAVeform:Y[:SCALe]:FIT	Runs auto-scale on the time domain display.
:DISPlay:OVIew:WAVeform:Y[:SCALe]:FULL	Sets the time domain display's vertical axis to the default full-scale.

Table 2–14: :DISPlay commands (Cont.)

Header	Description
:DISPlay:OVIew:WAVeform:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical value in the time domain display.
:DISPlay:OVIew:WAVeform:Y[:SCALe]:PDIVision(?)	Sets the vertical scale in the time domain display.
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:OFFSet(?)	Sets the minimum color-axis value of the spectrogram with zoom.
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:RANGe(?)	Sets the color-axis full-scale of the spectrogram with zoom.
:DISPlay:OVIew:ZOOM:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value of the spectrogram with zoom.
:DISPlay:OVIew:ZOOM:X[:SCALe]:SPAN(?)	Sets the horizontal full-scale of the spectrogram with zoom.
:DISPlay:OVIew:ZOOM:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical value of the spectrogram with zoom.
:DISPlay:OVIew:ZOOM:Y[:SCALe]:PLINe(?)	Sets the vertical scale of the spectrogram with zoom.
:DISPlay:PULSe:MVIew|:SVIew subgroup	The main view and subview related in the pulse measurements
:DISPlay:PULSe:MVIew:RESult:CHPower(?)	Determines whether to show channel power measurement results.
:DISPlay:PULSe:MVIew:RESult:DCYCle(?)	Determines whether to show duty cycle measurement results.
:DISPlay:PULSe:MVIew:RESult:EBWidth(?)	Determines whether to show EBW measurement results.
:DISPlay:PULSe:MVIew:RESult:FREQuency(?)	Determines whether to show frequency deviation measurement results.
:DISPlay:PULSe:MVIew:RESult:OBWidth(?)	Determines whether to show OBW measurement results.
:DISPlay:PULSe:MVIew:RESult:OORatio(?)	Determines whether to show on/off-ratio measurement results.
:DISPlay:PULSe:MVIew:RESult:PERiod(?)	Determines whether to show repetition interval measurement results.
:DISPlay:PULSe:MVIew:RESult:PHASe(?)	Determines whether to show pulse-pulse phase measurement results.
:DISPlay:PULSe:MVIew:RESult:PPOWer(?)	Determines whether to show peak power measurement results.
:DISPlay:PULSe:MVIew:RESult:RIPPle(?)	Determines whether to show pulse ripple measurement results.
:DISPlay:PULSe:MVIew:RESult:WIDTh(?)	Determines whether to show pulse width measurement results.
:DISPlay:PULSe:SVIew:FORMat(?)	Selects the display format of the subview.
:DISPlay:PULSe:SVIew:GUIDelines(?)	Determines whether to show the guidelines in the subview.
:DISPlay:PULSe:SVIew:RANGe(?)	Selects how to set the horizontal scale in the subview.
:DISPlay:PULSe:SVIew:RESult(?)	Selects how to show the result graph in the subview.
:DISPlay:PULSe:SVIew:SELect(?)	Selects a pulse to measure.
:DISPlay:PULSe:SPECtrum subgroup	The spectrum view related in the pulse measurements
:DISPlay:PULSe:SPECtrum:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value (left edge).
:DISPlay:PULSe:SPECtrum:X[:SCALe]:PDIVision(?)	Sets the horizontal scale (per division).
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:FIT	Runs the auto-scale.
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:FULL	Sets the vertical axis to the default full-scale value.
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical value (bottom).
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:PDIVision(?)	Sets the vertical scale (per division).
:DISPlay:PULSe:WAVeform subgroup	Time domain display related in the pulse measurements
:DISPlay:PULSe:WAVeform:X[:SCALe]:OFFSet(?)	Sets the minimum value of the horizontal axis (left edge).
:DISPlay:PULSe:WAVeform:X[:SCALe]:PDIVision(?)	Sets or queries the horizontal scale (per division).

Table 2–14: :DISPlay commands (Cont.)

Header	Description
:DISPlay:PULSe:WAVeform:Y[:SCALe]:FIT	Runs the auto-scale.
:DISPlay:PULSe:WAVeform:Y[:SCALe]:FULL	Sets the vertical axis to the default full-scale value.
:DISPlay:PULSe:WAVeform:Y[:SCALe]:OFFSet(?)	Sets the minimum value (bottom) of the vertical axis.
:DISPlay:PULSe:WAVeform:Y[:SCALe]:PDIVision(?)	Sets the vertical scale (per division).
:DISPlay:SPECtrum subgroup	Spectrum measurement related.
:DISPlay:SPECtrum:BMARker:STATe(?)	Turns on or off the band power marker.
:DISPlay:SPECtrum:GRATicule:GRID(?)	Determines how the graticule is displayed.
:DISPlay:SPECtrum:MLINe:AMPLitude:INTerval(?)	Sets the interval of the amplitude multi display lines.
:DISPlay:SPECtrum:MLINe:AMPLitude:OFFSet(?)	Sets the offset of the amplitude multi display lines.
:DISPlay:SPECtrum:MLINe:AMPLitude[:STATe](?)	Determines whether to show the amplitude multi display lines.
:DISPlay:SPECtrum:MLINe:ANNotation[:STATe](?)	Determines whether to show the readout of the multi display lines.
:DISPlay:SPECtrum:MLINe:FREQuency:INTerval(?)	Sets the interval of the frequency multi display lines.
:DISPlay:SPECtrum:MLINe:FREQuency:OFFSet(?)	Sets the offset of the frequency multi display line.
:DISPlay:SPECtrum:MLINe:FREQuency[:STATe](?)	Determines whether to show the frequency multi display lines.
:DISPlay:SPECtrum:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value (start frequency).
:DISPlay:SPECtrum:X[:SCALe]:PDIVision(?)	Sets the horizontal scale (span/div).
:DISPlay:SPECtrum:Y[:SCALe]:FIT	Runs auto-scale.
:DISPlay:SPECtrum:Y[:SCALe]:FULL	Sets the vertical axis to the default full-scale.
:DISPlay:SPECtrum:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical, or amplitude, value (bottom end).
:DISPlay:SPECtrum:Y[:SCALe]:PDIVision(?)	Sets the vertical, or amplitude, scale per division.
:DISPlay:TFRequency subgroup	3-dimensional view related.
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:OFFSet(?)	Sets the minimum color-axis value (bottom end) of the spectrogram.
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:RANGe(?)	Sets the scale of the spectrogram's color axis.
:DISPlay:TFRequency:SGRam:MLINe:ANNotation[:STATe](?)	Determines whether to show the readout of the multi display lines.
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:INTerval(?)	Sets the interval of the frequency multi display lines.
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:OFFSet(?)	Sets the offset of the frequency multi display lines.
:DISPlay:TFRequency:SGRam:MLINe:FREQuency[:STATe](?)	Determines whether to show the frequency multi display lines.
:DISPlay:TFRequency:SGRam:MLINe:TIME:INTerval(?)	Sets the interval of the time multi display lines.
:DISPlay:TFRequency:SGRam:MLINe:TIME:OFFSet(?)	Sets the offset of the time multi display lines.
:DISPlay:TFRequency:SGRam:MLINe:TIME[:STATe](?)	Determines whether to show the time multi display lines.
:DISPlay:TFRequency:SGRam:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal value (left end) of the spectrogram.
:DISPlay:TFRequency:SGRam:X[:SCALe]:SPAN(?)	Sets the horizontal full-scale (span) of the spectrogram.

Table 2–14: :DISPlay commands (Cont.)

Header	Description
:DISPlay:TFRequency:SGRam:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical value (bottom end) of the spectrogram.
:DISPlay:TFRequency:SGRam:Y[:SCALe]:PLINe(?)	Sets the vertical scale of the spectrogram.
:DISPlay[:VIEW] subgroup	General conditions about display.
:DISPlay[:VIEW]:BRIGhtness(?)	Sets the display brightness.
:DISPlay[:VIEW]:FORMat(?)	Selects the view display format.
:DISPlay:WAVeform subgroup	Time domain display related.
:DISPlay:WAVeform:X[:SCALe]:OFFSet(?)	Sets the minimum horizontal, or time, value (left end).
:DISPlay:WAVeform:X[:SCALe]:PDIVision(?)	Sets the horizontal, or time, scale per division.
:DISPlay:WAVeform:Y[:SCALe]:FIT	Runs auto-scale.
:DISPlay:WAVeform:Y[:SCALe]:FULL	Sets the vertical axis to the default full-scale.
:DISPlay:WAVeform:Y[:SCALe]:OFFSet(?)	Sets the minimum vertical, or amplitude, value (bottom end).
:DISPlay:WAVeform:Y[:SCALe]:PDIVision(?)	Sets the vertical, or amplitude, scale.

:FETCh Commands

The :FETCh commands retrieve the measurements from the data taken by the latest INITiate command.

If you want to perform a FETCh operation on fresh data, use the :READ commands, which acquire a new input signal and fetch the measurement results from that data.

Table 2–15: :FETCh commands

Header	Description
:FETCh:ADEMod:AM?	Returns the AM signal analysis results in time series.
:FETCh:ADEMod:AM:RESult?	Returns the AM signal analysis results.
:FETCh:ADEMod:FM?	Returns the FM signal analysis results in time series.
:FETCh:ADEMod:FM:RESult?	Returns the FM signal analysis results.
:FETCh:ADEMod:PM?	Returns the PM signal analysis results in time series.
:FETCh:ADEMod:PSPectrum?	Returns the spectrum data of the pulse spectrum measurement.
:FETCh:CCDF?	Returns the CCDF measurement results.
:FETCh:DISTribution:CCDF?	Returns the CCDF trace data.
:FETCh:OVIew?	Returns the maximum and minimum of waveform on the overview.
:FETCh:PULSe?	Returns the result of the pulse characteristics analysis.
:FETCh:PULSe:SPECtrum?	Returns the spectrum data of the frequency domain measurement.

Table 2–15: :FETCh commands (Cont.)

Header	Description
:FETCh:PULSe:TAMPlitude?	Returns the time domain amplitude data.
:FETCh:PULSe:TFRequency?	Returns the frequency deviation measurement results.
:FETCh:SPECtrum?	Returns spectrum waveform data.
:FETCh:SPECtrum:ACPower?	Returns the ACPR measurement results.
:FETCh:SPECtrum:CFRequency?	Returns the carrier frequency measurement results.
:FETCh:SPECtrum:CHPower?	Returns the channel power measurement results.
:FETCh:SPECtrum:CNRatio?	Returns the C/N measurement results.
:FETCh:SPECtrum:EBWidth?	Returns the emission bandwidth measurement results.
:FETCh:SPECtrum:OBWidth?	Returns the OBW measurement results.
:FETCh:SPECtrum:SPURious?	Returns the spurious signal measurement results.
:FETCh:TRANsient:FVTime?	Returns the frequency vs. time measurement results.
:FETCh:TRANsient:IQVTime?	Returns the I/Q level vs. time measurement results.
:FETCh:TRANsient:PVTime?	Returns the power vs. time measurement results.

:FORMat Commands

Define the data output format.

Table 2–16: :FORMat commands

Header	Description
:FORMat:BORDer(?)	Selects the byte order of output data.
:FORMat[:DATA](?)	Selects the data format for output.

:HCOPy Commands

Control hardcopy of the screen.

Table 2–17: :HCOPy commands

Header	Description
:HCOPy:BACKground	Selects the hardcopy background color.
:HCOPy:DESTination	Selects the hardcopy output destination.
:HCOPy[:IMMediate]	Outputs the hardcopy to the specified printer.

:INITiate Commands

Control data acquisition.

Table 2–18: :INITiate commands

Header	Description
:INITiate:CONTinuous(?)	Determines whether to acquire data continuously.
:INITiate[:IMMediate]	Starts data acquisition.
:INITiate:RESTart	Restarts data acquisition.

:INPut Commands

Control the characteristics of the signal input.

Table 2–19: :INPut commands

Header	Description
:INPut:ALEVel	Adjusts amplitude automatically for the best system performance.
:INPut:ATTenuation(?)	Sets the input attenuation.
:INPut:ATTenuation:AUTO(?)	Determines whether to set the input attenuation automatically.
:INPut:MIXer(?)	Sets the mixer level.
:INPut:MLEVel(?)	Sets the reference level.

:INSTrument Commands

Sets the measurement mode for the analyzer.

Table 2–20: :INSTrument commands

Header	Description
:INSTrument:CATalog?	Queries all the measurement modes that the analyzer has.
:INSTrument[:SELect]	Selects the measurement mode.

:MMEMory Commands

Manipulates files residing on the internal hard disk or floppy disk.

Table 2–21: :MMEMory commands

Header	Description
:MMEMory:COPY	Copies the contents of a file to another.
:MMEMory:DELete	Deletes a file.
:MMEMory:LOAD:CORRection	Loads the correction table from a file.
:MMEMory:LOAD:IQT	Loads the IQ data from a file.
:MMEMory:LOAD:STATe	Loads the analyzer settings from a file.
:MMEMory:LOAD:TRACe	Loads trace data from a file.
:MMEMory:NAME	Specifies the file name for hard copy output.
:MMEMory:STORe:CORRection	Stores an amplitude correction table in a file.
:MMEMory:STORe:IQT	Stores IQ data in a file.
:MMEMory:STORe:PULSe	Stores the pulse measurement results in a file.
:MMEMory:STORe:STATe	Stores the analyzer settings in a file.
:MMEMory:STORe:TRACe	Stores trace data in a file.

:PROGram Commands

Control macro programs.

Table 2–22: :PROGram commands

Header	Description
:PROGram:CATalog?	Queries the list of macro programs.
:PROGram[:SELected]:DELete[:SELected]	Deletes a macro program.
:PROGram[:SELected]:EXECute	Runs a macro program.
:PROGram[:SELected]:NAME(?)	Specifies a macro program.
:PROGram:NUMBer(?)	Sets numeric variables for a program.
:PROGram:STRing(?)	Sets character variables for a program.

:READ Commands

The :READ commands acquire an input signal once in the single mode and obtain the measurement results from that data.

If you want to fetch the measurement results from the data currently residing in the memory without acquiring the input signal, use the :FETCh commands.

Table 2–23: :READ commands

Header	Description
:READ:ADEMod:AM?	Returns the AM signal analysis results in time series.
:READ:ADEMod:AM:RESult?	Returns the AM signal analysis results.
:READ:ADEMod:FM?	Returns the FM signal analysis results in time series.
:READ:ADEMod:FM:RESult?	Returns the FM signal analysis results.
:READ:ADEMod:PM?	Returns the PM signal analysis results in time series.
:READ:ADEMod:PSPectrum?	Returns the spectrum data of the pulse spectrum measurement.
:READ:CCDF?	Returns the CCDF measurement results.
:READ:DISTribution:CCDF?	Returns the CCDF trace data.
:READ:OVIew?	Returns the maximum and minimum of waveform on the overview.
:READ:PULSe?	Returns the result of the pulse characteristics analysis.
:READ:PULSe:SPECtrum?	Returns the spectrum data of the frequency domain measurement.
:READ:PULSe:TAMPlitude?	Returns the time domain amplitude data.
:READ:PULSe:TFRequency?	Returns the frequency deviation measurement results.
:READ:SPECtrum?	Returns spectrum waveform data.
:READ:SPECtrum:ACPower?	Returns the ACPR measurement results.
:READ:SPECtrum:CFRequency?	Returns the carrier frequency measurement results.
:READ:SPECtrum:CHPower?	Returns the channel power measurement results.
:READ:SPECtrum:CNRatio?	Returns the C/N measurement results.
:READ:SPECtrum:EBWidth?	Returns the emission bandwidth measurement results.
:READ:SPECtrum:OBWidth?	Returns the OBW measurement results.
:READ:SPECtrum:SPURious?	Returns the spurious signal measurement results.
:READ:TRANsient:FVTime?	Returns the frequency vs. time measurement results.
:READ:TRANsient:IQVTime?	Returns the I/Q level vs. time measurement results.
:READ:TRANsient:PVTime?	Returns the power vs. time measurement results.

:SENSe Commands

Set the detailed measurement conditions.

Table 2–24: :SENSe commands

Header	Description
[:SENSe]:ACPower subgroup	ACPR measurement related.
[:SENSe]:ACPower:BANDwidth|BWIDth:ACHannel(?)	Sets the bandwidth of the next adjacent channel.
[:SENSe]:ACPower:BANDwidth|BWIDth:INTegration(?)	Sets the bandwidth of the main channel.
[:SENSe]:ACPower:CSPacing(?)	Sets the channel-to-channel spacing.
[:SENSe]:ACPower:FILTer:COEFficient(?)	Sets the filter factor.
[:SENSe]:ACPower:FILTer:TYPE(?)	Selects a filter.
[:SENSe]:ADEMod subgroup	Analog modulation analysis related.
[:SENSe]:ADEMod:BLOCk(?)	Sets the number of the block to be measured.
[:SENSe]:ADEMod:CARRier:OFFSet(?)	Sets the carrier frequency offset in the FM signal analysis.
[:SENSe]:ADEMod:CARRier:SEARch(?)	Determines whether to detect the FM carrier automatically.
[:SENSe]:ADEMod:FM:THReshold(?)	Sets the threshold level to determine a burst in the FM analysis.
[:SENSe]:ADEMod[:IMMediate]	Runs the analog modulation analysis.
[:SENSe]:ADEMod:LENGth(?)	Sets the length of the measurement range.
[:SENSe]:ADEMod:MODulation(?)	Selects the modulation.
[:SENSe]:ADEMod:OFFSet(?)	Sets the measurement start position.
[:SENSe]:ADEMod:PM:THReshold(?)	Sets the threshold level to determine a burst in the PM analysis.
[:SENSe]:AVERage subgroup	Averaging related.
[:SENSe]:AVERage:CLEar	Restarts the averaging from the beginning.
[:SENSe]:AVERage:COUNt(?)	Sets the number of averages.
[:SENSe]:AVERage[:STATe](?)	Turns on or off averaging.
[:SENSe]:AVERage:TCONtrol(?)	Selects the operation when the number of averages is reached.
[:SENSe]:BSIZe subgroup	Block size setting.
[:SENSe]:BSIZe(?)	Sets the block size.
CCDF subgroup	CCDF measurement related.
[:SENSe]:CCDF:BLOCk(?)	Sets the number of the block to be measured.
[:SENSe]:CCDF:CLEar	Restarts the measurement from the beginning.
[:SENSe]:CCDF:RMEasurement(?)	Clears the CCDF accumulator and restarts the measurement..
[:SENSe]:CCDF:THReshold(?)	Sets the threshold to include the samples in the CCDF calculation.
[:SENSe]:CFRequency subgroup	Carrier frequency measurement related.
[:SENSe]:CFRequency:CRESolution(?)	Sets the counter resolution.

Table 2–24: :SENSe commands (Cont.)

Header	Description
[:SENSe]:CHPower subgroup	Channel power measurement related.
[:SENSe]:CHPower:BANDwidth|BWIDth:INTegration(?)	Sets the channel bandwidth.
[:SENSe]:CHPower:FILTer:COEFficient(?)	Sets the filter roll-off rate.
[:SENSe]:CHPower:FILTer:TYPE(?)	Selects the filter.
[:SENSe]:CNRatio subgroup	Carrier-to-Noise (C/N) measurement related.
[:SENSe]:CNRatio:BANDwidth|BWIDth:INTegration(?)	Sets the measurement bandwidth.
[:SENSe]:CNRatio:BANDwidth|BWIDth:NOISe(?)	Sets the noise bandwidth.
[:SENSe]:CNRatio:FILTer:COEFficient(?)	Sets the filter roll-off rate.
[:SENSe]:CNRatio:FILTer:TYPE(?)	Selects the filter.
[:SENSe]:CNRatio:OFFSet(?)	Sets the offset frequency.
[:SENSe]:CORRection subgroup	Amplitude correction related.
[:SENSe]:CORRection:DATA(?)	Sets amplitude correction data.
[:SENSe]:CORRection:DELete	Deletes amplitude correction data.
[:SENSe]:CORRection:OFFSet[:MAGNitude](?)	Sets amplitude offset.
[:SENSe]:CORRection:OFFSet:FREQuency(?)	Sets frequency offset.
[:SENSe]:CORRection[:STATe](?)	Turns on or off amplitude correction.
[:SENSe]:CORRection:X:SPACing(?)	Selects scaling of the horizontal axis (frequency) for interpolation.
[:SENSe]:CORRection:Y:SPACing(?)	Selects scaling of the vertical axis (amplitude) for interpolation.
[:SENSe]:EBWidth subgroup	EBW measurement related.
[:SENSe]:EBWidth:XDB(?)	Sets the relative power from the peak for the measurement.
[:SENSe]:FEED subgroup	Input port related.
[:SENSe]:FEED	Selects the input port (RF, IQ, or calibration signal).
[:SENSe]:FREQuency subgroup	Frequency related.
[:SENSe]:FREQuency:BAND?	Queries the measurement frequency band.
[:SENSe]:FREQuency:CENTer(?)	Sets the center frequency.
[:SENSe]:FREQuency:CENTer:STEP:AUTO(?)	Determines whether to set the step size automatically by span.
[:SENSe]:FREQuency:CENTer:STEP[:INCRement](?)	Sets the step size of the center frequency.
[:SENSe]:FREQuency:CHANnel(?)	Selects a channel.
[:SENSe]:FREQuency:CTABle:CATalog?	Queries the available channel tables.
[:SENSe]:FREQuency:CTABle[:SELect](?)	Selects a channel table.
[:SENSe]:FREQuency:SPAN(?)	Sets the span.
[:SENSe]:FREQuency:STARt(?)	Sets the start frequency.
[:SENSe]:FREQuency:STOP(?)	Sets the stop frequency.

Table 2–24: :SENSe commands (Cont.)

Header	Description
[:SENSe]:OBWidth subgroup	OBW measurement related.
[:SENSe]:OBWidth:PERCent(?)	Sets the occupied bandwidth.
[:SENSe]:PULSe subgroup	Pulse characteristics analysis related
[:SENSe]:PULSe:BLOCk(?)	Sets the number of the block to measure.
[:SENSe]:PULSe:CHPower:BANDwidth|:BWIDth:INTegration(?)	Sets the channel bandwidth for the channel power measurement.
[:SENSe]:PULSe:CRESolution(?)	Sets the frequency measurement resolution.
[:SENSe]:PULSe:EBWidth:XDB(?)	Sets the level at which the EBW is measured.
[:SENSe]:PULSe:FFT:COEFficient(?)	Sets the roll-off ratio for the Nyquist FFT window.
[:SENSe]:PULSe:FFT:WINDow[:TYPE](?)	Selects the FFT window type.
[:SENSe]:PULSe:FILTer:BANDwidth|BWIDth(?)	Sets the bandwidth of the time measurement filter.
[:SENSe]:PULSe:FILTer:COEFficient(?)	Sets the a/BT value for the Gaussian measurement filter.
[:SENSe]:PULSe:FILTer:MEASuerment(?)	Selects the measurement filter for the time measurement.
[:SENSe]:PULSe:FREQuency:OFFSet(?)	Sets the frequency offset.
[:SENSe]:PULSe:FREQuency:RECovery(?)	Selects the frequency recovery.
[:SENSe]:PULSe[:IMMediate](?)	Runs calculation for acquired data.
[:SENSe]:PULSe:OBWidth:PERcent(?)	Sets OBW for the OBW measurement.
[:SENSe]:PULSe:PTOFfset(?)	Sets the time offset for the pulse-pulse phase measurement point.
[:SENSe]:PULSe:THReshold(?)	Sets the threshold level to detect pulses in acquired data.
[:SENSe]:ROSCillator subgroup	Reference oscillator related.
[:SENSe]:ROSCillator:SOURce(?)	Selects the reference oscillator.
[:SENSe]:SPECtrum subgroup	Spectrum related.
[:SENSe]:SPEctrum:AVERage:CLEar	Restarts the average process.
[:SENSe]:SPEctrum:AVERage:COUNt(?)	Sets the number of averages.
[:SENSe]:SPEctrum:AVERage[:STATe](?)	Turns on or off averaging.
[:SENSe]:SPEctrum:AVERage:TYPE(?)	Selects the average type.
[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution](?)	Sets the resolution bandwidth.
[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution]:AUTO(?)	Determines whether to automatically set the resolution bandwidth.
[:SENSe]:SPECtrum:BANDwidth|BWIDth:STATe(?)	Turns on or off the resolution bandwidth calculation process.
[:SENSe]:SPECtrum:DETector[:FUNCtion](?)	Determines how the trace is compressed.
[:SENSe]:SPECtrum:FILTer:COEFficient(?)	Sets the filter roll-off rate.
[:SENSe]:SPECtrum:FILTer:TYPE(?)	Selects the filter.
[:SENSe]:SPECtrum:FFT:ERESolution(?)	Determines whether to enable the extended resolution.
[:SENSe]:SPECtrum:FFT:LENGth(?)	Sets the number of FFT sample points.
[:SENSe]:SPECtrum:FFT:STARt(?)	Sets the time interval between 1024-point overlapped FFT frames.
[:SENSe]:SPECtrum:FFT:WINDow[:TYPE](?)	Selects a FFT window.

Table 2–24: :SENSe commands (Cont.)

Header	Description
[:SENSe]:SPECtrum:FRAMe(?)	Sets the frame number for the spectrum measurement.
[:SENSe]:SPECtrum:MEASurement(?)	Runs a selected measurement item.
[:SENSe]:SPECtrum:ZOOM:BLOCk(?)	Sets the number of the block to zoom.
[:SENSe]:SPECtrum:ZOOM:FREQuency:CENTer(?)	Sets the center frequency of a zoomed area.
[:SENSe]:SPECtrum:ZOOM:FREQuency:WIDTh(?)	Sets the frequency width of a zoomed area.
[:SENSe]:SPECtrum:ZOOM:LENGth(?)	Sets the time length of a zoomed area.
[:SENSe]:SPECtrum:ZOOM:OFFSet(?)	Sets the starting point of a zoomed area.
[:SENSe]:SPURious subgroup	Spurious signal measurement related.
[:SENSe]:SPURious[:THReshold]:EXCursion(?)	Sets the spurious excursion level.
[:SENSe]:SPURious[:THReshold]:IGNore(?)	Sets an area to ignore spurious.
[:SENSe]:SPURious[:THReshold]:SIGNal(?)	Sets the carrier criterion level.
[:SENSe]:SPURious[:THReshold]:SPURious(?)	Sets the spurious criterion level.
[:SENSe]:TRANsient subgroup	Time analysis related.
[:SENSe]:TRANsient:BLOCk(?)	Sets the number of the block to be measured.
[:SENSe]:TRANsient[:IMMediate]	Starts a time characteristic analysis.
[:SENSe]:TRANsient:ITEM(?)	Selects a measurement item.
[:SENSe]:TRANsient:LENGth(?)	Sets the length of the measurement range.
[:SENSe]:TRANsient:OFFSet(?)	Sets the measurement start position.

:STATus Commands

Control registers defined in the SCPI status reporting structure.

Table 2–25: :STATus commands

Header	Description
:STATus:OPERation:CONDition?	Queries the contents of the OCR.
:STATus:OPERation:ENABle(?)	Sets the mask for the OENR.
:STATus:OPERation[:EVENt]?	Queries the contents of the OEVR.
:STATus:OPERation:NTRansition(?)	Sets the value of the negative transition filter.
:STATus:OPERation:PTRansition(?)	Sets the value of the positive transition filter.
:STATus:PRESet	Presets a status byte.
:STATus:QUEStionable:CONDition?	Queries the contents of the QCR.
:STATus:QUEStionable:ENABle(?)	Sets the mask for the OENR.
:STATus:QUEStionable[:EVENt]?	Queries the contents of the QER.

Table 2–25: :STATus commands (Cont.)

Header	Description
:STATus:QUEStionable:NTRansition(?)	Sets the value of the negative transition filter.
:STATus:QUEStionable:PTRansition(?)	Sets the value of the positive transition filter.

:SYSTem Commands

Set the system parameters and query system information.

Table 2–26: :SYSTem commands

Header	Description
:SYSTem:DATE(?)	Sets the current date.
:SYSTem:ERRor:ALL?	Queries all the error or event information.
:SYSTem:ERRor:CODE:ALL?	Queries all the error or event codes.
:SYSTem:ERRor:CODE[:NEXT]?	Queries the latest error or event codes.
:SYSTem:ERRor:COUNt?	Queries the number of errors or events.
:SYSTem:ERRor[:NEXT]?	Queries the latest error or event information.
:SYSTem:KLOCk(?)	Determine whether to lock or unlock the front panel keys.
:SYSTem:OPTions?	Queries optional information.
:SYSTem:PRESet	Presets the analyzer.
:SYSTem:TIME(?)	Sets the current time.
:SYSTem:VERSion?	Queries the version of the SCPI.

:TRACe Commands

Set up display of Trace 1 and 2.

Table 2–27: :TRACe commands

Header	Description
:TRACe<x>|:DATA<x>:AVERage:CLEar	Restarts trace averaging.
:TRACe<x>|:DATA<x>:AVERage:COUNt(?)	Sets the number of traces to combine for averaging.
:TRACe<x>|:DATA<x>:DDETector(?)	Selects the display detector.
:TRACe<x>|:DATA<x>:MODE(?)	Selects the way to display the traces.

:TRIGger Commands

Set up the trigger.

Table 2–28: :TRIGger commands

Header	Description
:TRIGger[:SEQuence]:LEVel:IF(?)	Sets the IF trigger level.
:TRIGger[:SEQuence]:MODE(?)	Selects the trigger mode.
:TRIGger[:SEQuence]:MPOSition?	Queries the trigger occurrence point in one block data on the memory.
:TRIGger[:SEQuence]:OPOSition?	Queries the trigger output point.
:TRIGger[:SEQuence]:POSition(?)	Sets the trigger position.
:TRIGger[:SEQuence]:SAVE:COUNt[:STATe](?)	Determines whether to limit the number of times that data is saved.
:TRIGger[:SEQuence]:SAVE:COUNt:MAXimum(?)	Sets a limit on the number of times that data is saved.
:TRIGger[:SEQuence]:SAVE[:STATe](?)	Determines whether to enable or disable the Save-on-Trigger function.
:TRIGger[:SEQuence]:SLOPe(?)	Selects the trigger slope.
:TRIGger[:SEQuence]:SOURce(?)	Selects the trigger source.

:UNIT Commands

Specify fundamental units for measurement.

Table 2–29: :UNIT commands

Header	Description
:UNIT:ANGLe(?)	Specifies the fundamental unit of angle.

General Programming Procedure

You should generally use the following procedure to script a program:

1. *Setting the measurement mode*
   Using an :INSTrument command, select a measurement mode to set the basic conditions.
   [Example] `:INSTrument:SELect "SANORMAL"`
   Selects the normal spectrum analysis mode to set the basic conditions.

2. *Setting the measurement item*
   Using a :CONFigure command, select a measurement item to set up the analyzer to the defaults.
   [Example] `:CONFigure:SPECtrum:CHPower`
   Sets up the analyzer to the channel power measurement defaults.

3. *Detailed settings*
   Use :SENSe commands to set details for the measurement session.
   [Example] `:SENSe:CHPower:BWIDth:INTegration 3MHz`
   Sets the channel power measurement range to 3 MHz.

4. *Acquiring data*
   Use an :INITiate or :ABORt command to initiate or stop data acquisition.
   [Example] `:INITiate:CONTinuous ON`
   Initiates data acquisition in continuous mode.

   To save or load the acquired data and settings, use an :MMEMory command.
   [Example] `:MMEMory:STORe:IQT "DATA1"`
   Saves the acquired data in file DATA1.IQT.

5. *Obtaining the measurement results*
   Use a :FETCh or :READ command to obtain the measurement results.
   [Example] `:FETCh:SPECtrum:CHPower?`
   Returns the channel power measurement results.

6. *Display*
   Use :DISPlay commands to set the display-related conditions.
   [Example] `:DISPlay:SPECtrum:X:SCALe:OFFSet 800MHz`
   Sets 800 MHz for the minimum (left) edge of the horizontal axis in the spectrum view.

Also refer to Chapter 4, *Programming Examples*.
Appendix C lists the default settings of the commands.

IEEE Common Commands

This section details the IEEE common commands.

Command Tree

Header	Parameter
*CAL?	
*CLS	
*ESE	<numeric_value>
*ESR?	
*IDN?	
*OPC	
*OPT?	
*RST	
*SRE	<numeric_value>
*STB?	
*TRG	
*TST?	
*WAI	

*CAL? (Query Only)

Runs the following three calibrations and returns the results indicating whether they have ended normally.

RF gain calibration
Center offset calibration
DC offset calibration (if the measurement frequency band is the baseband)

This command is equivalent to the :CALibration[:ALL]? query command.

***NOTE**. The entire calibration process takes several minutes to several dozen minutes. Wait for a response from a *CAL query. Every command you attempt to send during this period is rejected.*

Syntax

```
*CAL?
```

Arguments

None

Returns

<NR1>

0 indicates a normal end. For details of the error codes, refer to page 3–17.

Measurement Modes

All

Examples

```
*CAL?
```

runs a calibration and might return 0, indicating that the calibration has ended normally.

Related Commands

```
:CALibration[:ALL]
```

*CLS (No Query Form)

Clears all the event status registers and queues used in the status/event reporting structure. Refer to Section 3, *Status and Events*, for the register information.

Syntax `*CLS`

Arguments None

Measurement Modes All

Examples `*CLS`
clears all the event status registers and queues.

Related Commands `*ESE`, `*ESR`, `*SRE`, `*STB?`

*ESE (?)

Sets or queries the value of the Event Status Enable Register (ESER) used in the status/event reporting structure. Refer to Section 3, *Status and Events*, for the register information.

Syntax `*ESE <value>`

`*ESE?`

Arguments `<value>::=<NR1>` is a value in the range from 0 through 255. The binary bits of the ESER are set according to this value.

Measurement Modes All

Examples `*ESE 145`
sets the ESER to binary 10010001, which enables the PON, EXE, and OPC bits.

`*ESE?`
might return the string `*ESE 184`, showing that the ESER contains the binary value 10111000.

Related Commands `*CLS`, `*ESR`, `*SRE`, `*STB?`

*ESR? (Query Only)

Sets or queries the contents of the Standard Event Status Register (SESR) used in the status/event reporting structure. The SESR is cleared after being read. Refer to Section 3, *Status and Events*, for the register information.

Syntax `*ESR?`

Arguments None

Returns `<NR1>` representing the contents of the SESR by a 0 to 255 decimal number.

Measurement Modes All

Examples `*ESR?`
might return the value 213, showing that the SESR contains binary 11010101.

Related Commands `*CLS`, `*ESE?`, `*SRE`, `*STB?`

*IDN? (Query Only)

Returns the analyzer's identification code.

Syntax `*IDN?`

Arguments None

Returns The analyzer identification code in the following format:

```
TEKTRONIX,RSA220XA,<serial_number>,<firmware_version>
```

Where
`TEKTRONIX` indicates that the manufacturer is Tektronix.
`RSA220XA` is RSA2203A or RSA2208A, depending on the model.
`<serial_number>` is the serial number.
`<firmware_version>` is the firmware version.

Measurement Modes All

Examples `*IDN?`
might return `TEKTRONIX,RSA2208A,J300101,1.20` as the analyzer's identification code.

*OPC (?)

Generates the operation complete message in the Standard Event Status Register (SESR) when all pending operations finish. The *OPC? query places the ASCII character "1" into the output queue when all pending operations are finished. The *OPC? response is not available to read until all pending operations finish.

The *OPC command allows you to synchronize the operation of the analyzer with your application program. Refer to *Synchronizing Execution* on page 3–14 for the details.

Syntax `*OPC`

`*OPC?`

Arguments None

Measurement Modes All

*OPT? (Query Only)

Queries the options installed in the analyzer.

Syntax

```
*OPT?
```

Arguments

None

Returns

The numbers of all the options installed in the analyzer, separated by commas. If no options have been installed, 0 is returned.

Measurement Modes

All

Examples

```
*OPT?
```

might return 05,10,12, indicating that Option 05, 10, and 12 are currently installed in the analyzer.

*RST (No Query Form)

Restores the analyzer to the factory default settings. For the actual settings, refer to *Appendix C: Factory Initialization Settings*. This command is equivalent to a pair of commands :SYSTem:PRESet and *CLS that run successively.

The *RST command does not alter the following:

- The state of the IEEE Std 488.1–1987 interface.
- The selected IEEE Std 488.1–1987 address of the analyzer.
- Measurement mode selected with the :INSTrument[:SELect] command
- Calibration data that affect device specifications.
- The Output Queue.
- The Service Request Enable Register setting.
- The Standard Event Status Enable Register setting.
- The Power-on status clear flag setting.
- Stored settings.

Syntax `*RST`

Arguments None

Measurement Modes All

Examples `*RST`
resets the analyzer.

Related Commands `*CLS`, `:INSTrument[:SELect]`, `:SYSTem:PRESet`

*SRE (?)

Sets or queries the value of the Service Request Enable Register (SRER) used in the status/event reporting structure. Refer to Section 3, *Status and Events*, for the register information.

Syntax

```
*SRE <value>
```

```
*SRE?
```

Arguments

<value>::=<NR1> is a value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution error.

Measurement Modes

All

Examples

```
*SRE 48
```

sets binary 00110000 in the SRER's bits:

```
*SRE?
```

might return 32, indicating that binary value 00100000 has been set in the SRER's bits.

Related Commands

*CLS, *ESE, *ESR?, *STB?

*STB? (Query Only)

Returns the contents of the Status Byte Register (SBR) in the status/event reporting structure using the Master Summary Status (MSS) bit.
Refer to Section 3, *Status and Events*, for the register information.

Syntax `*STB?`

Arguments None

Returns `<NR1>` representing the contents of the SBR as a decimal number.

Measurement Modes All

Examples `*STB?`
might return 96, indicating that the SBR contains binary 0110 0000.

Related Commands `*CLS, *ESE, *ESR?, *SRE`

*TRG (No Query Form)

Generates a trigger signal.
This command is equivalent to the :INITiate[:IMMediate] command.

Syntax `*TRG`

Arguments None

Measurement Modes All

Examples `*TRG`
generates a trigger signal.

Related Commands `:INITiate[:IMMediate]`

*TST? (Query Only)

Runs a self test and returns the result.

***NOTE**. The analyzer does not run any self test. It returns 0 whenever a *TST command is sent.*

Syntax `*TST?`

Arguments None

Returns `<NR1>`. Always 0.

Measurement Modes All

Related Commands `*CAL?`, `CALibration[:ALL]`

*WAI (No Query Form)

Prevents the analyzer from executing further commands or queries until all pending operations finish. This command allows you to synchronize the operation of the analyzer with your application program. For the details, refer to *Synchronizing Execution* on page 3–14.

Syntax `*WAI`

Arguments None

Measurement Modes All

Related Commands `*OPC`

:ABORt Commands

Resets the trigger system and related actions such as data acquisition and measurement.

Command Tree

Header	Parameter
:ABORt	

:ABORt (No Query Form)

Resets the trigger system and related actions such as data acquisition and measurement.

***NOTE**. You must have acquired data using the :INITiate:CONTinuous command (refer to page 2–180) before you can execute the :ABORt command.*

The command function depends on the acquisition mode as follows.

For single acquisition mode:
The :ABORt command forcibly stops data acquisition.

To stop the acquisition because the trigger does not occur in the single mode, send this command:

```
:INITiate:CONTinuous OFF
```

For continuous acquisition mode:
The :ABORt command initiates a new session of data acquisition in the continuous mode.

To stop the acquisition in the continuous mode, send this command:

```
:INITiate:CONTinuous OFF
```

Syntax `:ABORt`

Arguments None

Measurement Modes All

Examples `:ABORt`
resets the trigger system and related actions such as data acquisition and measurement.

Related Commands `:INITiate:CONTinuous`

:CALCulate Commands

The :CALCulate commands control the marker and the display line. The views are identified with :CALCulate<x> in the command header (see Figure 2–5).

:CALCulate1: View 1
:CALCulate2: View 2 (NOTE: currently not used)
:CALCulate3: View 3
:CALCulate4: View 4

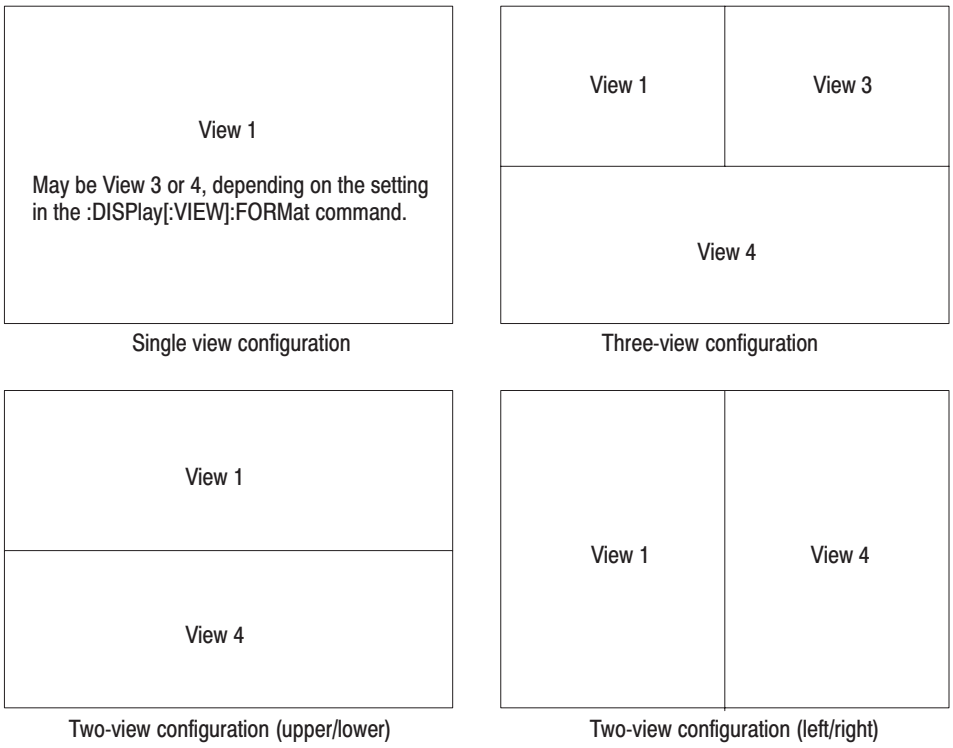


Figure 2–5: View number assignments

For details on the marker and the display line, refer to the *RSA2203A and RSA2208A User Manual*.

Command Tree

Header	Parameter
:CALCulate<x>	
:DLINe<y>	<numeric_value>
:STATe	<boolean>
:MARKer<y>	
:AOFF	
:MAXimum	
:MODE	POSition | DELTa
:PEAK	
:HIGHer	
:LEFT	
:LOWer	
:RIGHt	
:PTHReshold	<numeric_value>
:ROFF	
[:SET]	
:CENTer	
:MEASurement	
:RCURsor	
[:STATe]	<boolean>
:TOGGle	
:TRACe	MAIN | SUB
:X	<numeric_value>
:Y	<numeric_value>
:VLINe<y>	<numeric_value>
:STATe	<boolean>

:CALCulate<x>:DLINe<y> (?)

Sets or queries the vertical position of the horizontal line.

Syntax

```
:CALCulate<x>:DLINe<y> <value>
:CALCulate<x>:DLINe<y>?
```

Arguments

<value>::=<NRf> specifies the vertical position of the horizontal line. Range: –200 to +100 dBm

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:CALCulate1:DLINe1 -20
```

positions Horizontal Line 1 at –20 dBm in View 1.

Related Commands

:CALCulate<x>:DLINe<y>:STATe

:CALCulate<x>:DLINe<y>:STATe (?)

Determines whether to turn on or off the horizontal line.

Syntax

```
:CALCulate<x>:DLINe<y>:STATe { OFF | ON | 0 | 1 }
:CALCulate<x>:DLINe<y>:STATe?
```

Arguments

OFF or 0 hides the horizontal line.

ON or 1 shows the horizontal line.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:CALCulate1:DLINe2:STATe 1
```

shows Horizontal Line 2 in View 1.

:CALCulate<x>:MARKer<y>:AOFF (No Query Form)

Turns off all the markers of all the traces in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:AOFF`

Arguments None

Measurement Modes All

Examples `:CALCulate1:MARKer1:AOFF`
turns off all the markers of all the traces in View 1.

:CALCulate<x>:MARKer<y>:MAXimum (No Query Form)

Positions the marker at the maximum point on the trace in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:MAXimum`

Arguments None

Measurement Modes All

Examples `:CALCulate1:MARKer1:MAXimum`
positions the marker at the maximum point on the trace in View 1.

:CALCulate<x>:MARKer<y>:MODE (?)

Selects or queries the marker mode (position or delta) in the specified view.

Syntax

```
:CALCulate<x>:MARKer<y>:MODE { POSition | DELTa }
:CALCulate<x>:MARKer<y>:MODE?
```

Arguments

POSition selects the position marker mode, in which the marker measurement is performed without the reference cursor. It works the same for both <y>=1 and 2.

DELTa selects the delta marker mode, in which the marker measurement is performed with the reference cursor. The reference cursor is placed at the position of the specified marker.

Measurement Modes

All

Examples

```
:CALCulate1:MARKer1:MODE DELTa
```

selects the delta marker mode in View 1.

:CALCulate<x>:MARKer<y>:PEAK:HIGHer (No Query Form)

Moves the marker higher in amplitude to the next peak in the specified view.

Syntax

```
:CALCulate<x>:MARKer<y>:PEAK:HIGHer
```

Arguments

None

Returns

If no peak exists, the error message “No Peak Found Error (202)” is returned.

Measurement Modes

All

Examples

```
:CALCulate1:MARKer1:PEAK:HIGHer
```

moves Marker 1 higher in amplitude to the next peak in View 1.

:CALCulate<x>:MARKer<y>:PEAK:LEFT (No Query Form)

Shifts the marker to the next peak on the left in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:PEAK:LEFT`

Arguments None

Returns If no peak exists, the error message “No Peak Found Error (202)” is returned.

Measurement Modes All

Examples `:CALCulate1:MARKer1:PEAK:LEFT`
shifts the marker to the next peak on the left in View 1.

:CALCulate<x>:MARKer<y>:PEAK:LOWer (No Query Form)

Moves the marker lower in amplitude to the next peak in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:PEAK:LOWer`

Arguments None

Returns If no peak exists, error message “No Peak Found Error (202)” is returned.

Measurement Modes All

Examples `:CALCulate1:MARKer1:PEAK:LOWer`
moves Marker 1 lower in amplitude to the next peak in View 1.

:CALCulate<x>:MARKer<y>:PEAK:RIGHt (No Query Form)

Shifts the marker to the next peak on the right in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:PEAK:RIGHt`

Arguments None

Returns If no peak exists, the error message "No Peak Found Error (202)" is returned.

Measurement Modes All

Examples `:CALCulate1:MARKer1:PEAK:RIGHt`
shifts the marker to the next peak on the right in View 1.

:CALCulate<x>:MARKer<y>:PTHReshold (?)

Sets or queries the horizontal minimum jump of the marker for peak search in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:PTHReshold <value>`

`:CALCulate<x>:MARKer<y>:PTHReshold?`

Arguments `<value>::=<NRf>` sets the minimum jump of the marker for peak search. Range: 1% to 20% of the span setting.

Measurement Modes All

Examples `:CALCulate1:MARKer1:PTHReshold 10kHz`
sets the minimum jump of Marker 1 to 10 kHz for peak search.

:CALCulate<x>:MARKer<y>:ROFF (No Query Form)

Turns off the reference cursor in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:ROFF`

Arguments None

Measurement Modes All

Examples `:CALCulate1:MARKer1:ROFF`
turns off the reference cursor in View 1.

Related Commands `:CALCulate<x>:MARKer<y>[:SET]:RCURsor`

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

Sets the center frequency to the value at the marker position in the specified view.

Syntax `:CALCulate<x>:MARKer<y>[:SET]:CENTer`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CALCulate1:MARKer1:SET:CENTer`
sets the center frequency to the value at the marker position in View 1.

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

Defines the measurement position using the marker(s) in the specified view.

***NOTE**. This command is available in a view that represents time along the horizontal axis.*

The function varies between the marker modes as follows:

- *For the position marker mode:*
  Sets the current position of the specified marker to the measurement start position.
- *For the delta marker mode:*
  Sets the current positions of the specified marker and the reference cursor to the measurement start and stop positions.

The marker mode is selected with the :CALCulate<x>:MARKer<y>:MODE command (refer to page 2–49).

Syntax `:CALCulate<x>:MARKer<y>[:SET]:MEASurement`

Arguments None

Measurement Modes `DEMADEM`, `TIMCCDF`, `TIMTRAN`

Examples `:CALCulate1:MARKer1:SET:MEASurement`
defines the measurement position using the marker in View 1.

Related Commands `:CALCulate<x>:MARKer<y>:MODE`

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

Displays the reference cursor at the marker position in the specified view.

Syntax `:CALCulate<x>:MARKer<y>[:SET]:RCURsor`

Arguments None

Measurement Modes All

Examples `:CALCulate1:MARKer1:SET:RCURsor`
displays the reference cursor in View 1.

Related Commands `:CALCulate<x>:MARKer<y>:ROFF`

:CALCulate<x>:MARKer<y>[:STATe] (?)

Determines whether to turn on or off the marker(s) in the specified view.

Syntax `:CALCulate<x>:MARKer<y>[:STATe] { OFF | ON | 0 | 1 }`

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

Arguments `OFF` or `0` hides the marker(s). If you have selected the delta marker mode, both the main and delta markers will be turned off.

`ON` or `1` shows the marker(s). If you have selected the delta marker mode, both the main and delta markers will be turned on.

To select a marker mode, use :CALCulate<x>:MARKer<y>:MODE.

Measurement Modes All

Examples `:CALCulate1:MARKer1:STATe ON`
enables Marker 1 in View 1.

Related Commands `:CALCulate<x>:MARKer<y>:MODE`

:CALCulate<x>:MARKer<y>:TOGGle (No Query Form)

Replaces the marker and the reference cursor with each other in the specified view.

Syntax `:CALCulate<x>:MARKer<y>:TOGGle`

Arguments None

Measurement Modes All

Examples `:CALCulate1:MARKer1:TOGGle`
replaces Marker 1 and the reference cursor with each other in View 1.

:CALCulate<x>:MARKer<y>:TRACe (?)

Selects the trace to place the marker in the specified view.

The query command returns the name of the trace on which the marker is currently placed.

Syntax `:CALCulate<x>:MARKer<y>:TRACe { MAIN | SUB }`

`:CALCulate<x>:MARKer<y>:TRACe?`

Arguments `MAIN` places the specified marker on Trace 1 (displayed in yellow on screen).

`SUB` places the specified marker on Trace 2 (displayed in green on screen).

Measurement Modes All

Examples `:CALCulate1:MARKer1:TRACe SUB`
places Marker 1 on Trace 2 in View 1.

:CALCulate<x>:MARKer<y>:X (?)

Sets or queries the horizontal position of the marker in the specified view.

Syntax

```
:CALCulate<x>:MARKer<y>:X <param>
:CALCulate<x>:MARKer<y>:X?
```

Arguments

<param>::=<NRf> specifies the horizontal marker position.

The parameter value is different between the marker modes as follows:

- *For the position marker mode:*
  Sets the absolute position of the specified marker.
- *For the delta marker mode:*
  Sets the relative position of the specified marker from the reference cursor.

The marker mode is selected with the :CALCulate<x>:MARKer<y>:MODE command (refer to page 2–49).

The valid setting range depends on the display format. Refer to Table D–1 in *Appendix D*.

Measurement Modes

All

Examples

```
:CALCulate1:MARKer1:X 800MHz
```

places Marker 1 at 800 MHz in View 1 when the horizontal axis represents frequency.

Related Commands

:CALCulate<x>:MARKer<y>:MODE

:CALCulate<x>:MARKer<y>:Y (?)

Sets or queries the vertical position of the marker in the specified view.

***NOTE**. The setting command is valid in the spectrogram view displayed in the Real Time S/A (real-time spectrum analysis) mode and in the overview of the Demod (modulation analysis) and the Time (time analysis) modes. If the command is executed in other views, the error message "Execution Error" (–200) is returned. The query is available in all views.*

Syntax

```
:CALCulate<x>:MARKer<y>:Y <param>
:CALCulate<x>:MARKer<y>:Y?
```

Arguments

`<param>::=<NRf>` specifies the vertical marker position.

The parameter value is different between the marker modes as follows:

- *For the position marker mode:*
  Sets the absolute position of the specified marker.
- *For the delta marker mode:*
  Sets the relative position of the specified marker from the reference cursor.

The marker mode is selected with the :CALCulate<x>:MARKer<y>:MODE command (refer to page 2–49).

For the setting range, refer to Table D–1 in *Appendix D*.

Measurement Modes

`SARTIME`, `DEMADEM`, `TIMCCDF`, and `TIMTRAN` for setting. All modes for query.

Examples

```
:CALCulate1:MARKer1:Y -20
```
places the first marker at frame #–20 in View 1 (spectrogram).

```
:CALCulate2:MARKer1:Y?
```
might return `-34.28` indicating the first marker readout is –34.28 dBm in View 2 (spectrum).

Related Commands

`:CALCulate<x>:MARKer<y>:MODE`

:CALCulate<x>:VLINe<y> (?)

Sets or queries the horizontal position of the vertical line.

Syntax

```
:CALCulate<x>:VLINe<y> <value>
:CALCulate<x>:VLINe<y>?
```

Arguments

<value>::=<NRf> specifies the horizontal position of the vertical line. Range: 0 Hz to 3 GHz for RSA2203A, or 8 GHz for RSA2208A.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:CALCulate1:VLINe1 800MHz
```

sets the horizontal position of Vertical Line 1 to 800 MHz.

Related Commands

:CALCulate<x>:VLINe<y>:STATe

:CALCulate<x>:VLINe<y>:STATe (?)

Determines whether to turn on or off the vertical line.

Syntax

```
:CALCulate<x>:VLINe<y>:STATe { OFF | ON | 0 | 1 }
:CALCulate<x>:VLINe<y>:STATe?
```

Arguments

OFF or 0 hides the vertical line.

ON or 1 shows the vertical line.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:CALCulate1:VLINe1:STATe ON
```

shows Vertical Line 1 in View 1.

:CALibration Commands

The :CALibration commands run calibrations on the analyzer.
For details on calibrations, refer to the *RSA2203A and RSA2208A User Manual*.

Command Tree

Header	Parameter
:CALibration	
[:ALL]	
:AUTO	
:DATA	
:DEFault	
:OFFSet	
:BASebanddc	
:CENTer	
:RF	

:CALibration[:ALL] (?)

Runs the following three calibrations:

RF gain calibration
Center offset calibration
DC offset calibration (if the measurement frequency band is the baseband)

The :CALibration[:ALL]? query command runs these calibrations and returns the results. This command is equivalent to the *CAL? query command.

Syntax

```
:CALibration[:ALL]
```

```
:CALibration[:ALL]?
```

Arguments

None

Returns

<NR1>

0 indicates a normal end. For details of the error codes, refer to page 3–17.

Measurement Modes

All

Examples

```
:CALibration:ALL
```

runs all calibrations.

Related Commands

*CAL?

:CALibration:AUTO (?)

Determines whether to run the RF gain calibration automatically.

Syntax

```
:CALibration:AUTO { OFF | ON | 0 | 1 }
:CALibration:AUTO?
```

Arguments

OFF or 0 specifies that the analyzer does not run the RF gain calibration automatically. Use the :CALibration:RF command to run the RF gain calibration.

ON or 1 specifies that the analyzer runs the RF gain calibration automatically.

Measurement Modes

All

Examples

```
:CALibration:AUTO ON
```
specifies that the analyzer runs the RF gain calibration automatically.

Related Commands

:CALibration:RF

:CALibration:DATA:DEFault (No Query Form)

Restores the calibration data to the factory defaults.

Syntax

```
:CALibration:DATA:DEFault
```

Arguments

None

Measurement Modes

All

Examples

```
:CALibration:DATA:DEFault
```
restores the calibration data to the factory defaults.

:CALibration:OFFSet:BASebanddc (?)

Option 05 Only

Runs the baseband DC offset calibration. The query version of this command runs the calibration and, if it ends normally, returns 0.

***NOTE**. This command is available when the analyzer operates in the baseband (DC to 20 MHz). The frequency setting must satisfy the following condition: (center frequency) + (span)/2 ≤ 17.5 MHz*

Syntax

```
:CALibration:OFFSet:BASebanddc
:CALibration:OFFSet:BASebanddc?
```

Arguments None

Returns <NR1>

0 indicates a normal end. For details of the error codes, refer to page 3–17.

Measurement Modes All

Examples

```
:CALibration:OFFSet:BASebanddc
```

runs the baseband DC offset calibration.

:CALibration:OFFSet:CENTer (?)

Runs the center offset calibration. The query version of this command runs the calibration and, if it ends normally, returns 0.

Syntax

```
:CALibration:OFFSet:CENTer
:CALibration:OFFSet:CENTer?
```

Arguments None

Returns <NR1>

0 indicates a normal end. For details of the error codes, refer to page 3–17.

Measurement Modes All

Examples

```
:CALibration:OFFSet:CENTer
```

runs the center offset calibration.

:CALibration:RF (?)

Runs the RF gain calibration. The query version of this command runs the calibration and, if it ends normally, returns 0.

Syntax

```
:CALibration:RF
```

```
:CALibration:RF?
```

Arguments

None

Returns

<NR1>

0 indicates a normal end. For details of the error codes, refer to page 3–17.

Measurement Modes

All

Examples

```
:CALibration:RF
```

runs the RF gain calibration.

Related Commands

:CALibration:AUTO

:CONFigure Commands

The :CONFigure commands set up the analyzer to the default settings for the specified measurement.

Command Tree

Header	Parameter
:CONFigure	
:ADEMod	
:AM	
:FM	
:PM	
:PSPectrum	
:CCDF	
:OVIew	
:PULSe	
:SPECtrum	
:ACPower	
:CFRequency	
:CHPower	
:CNRatio	
:EBWidth	
:OBWidth	
:SPURious	
:TFRequency	
:RTIMe	
:SGRam	
:TRANsient	
:FVTime	
:IQVTime	
:PVTime	

***NOTE**. Data acquisition stops on completion of a :CONFigure command. The following each command description shows the front-panel key operation equivalent to running the command except data acquisition control.*

:CONFigure:ADEMod:AM (No Query Form)

Sets up the analyzer to the default settings for AM signal analysis.
Running this command is equivalent to pressing the following front panel keys:

DEMOD key → **Analog Demod** side key → **PRESET** key → **MEASURE** key → **AM Demod** side key

Syntax `:CONFigure:ADEMod:AM`

Arguments None

Measurement Modes `DEMADEM`

Examples `:CONFigure:ADEMod:AM`
sets up the analyzer to the default settings for AM signal analysis.

Related Commands `:INSTrument[:SELect]`

:CONFigure:ADEMod:FM (No Query Form)

Sets up the analyzer to the default settings for FM signal analysis.
Running this command is equivalent to pressing the following front panel keys:

DEMOD key → **Analog Demod** side key → **PRESET** key → **MEASURE** key → **FM Demod** side key

Syntax `:CONFigure:ADEMod:FM`

Arguments None

Measurement Modes `DEMADEM`

Examples `:CONFigure:ADEMod:FM`
sets up the analyzer to the default settings for FM signal analysis.

Related Commands `:INSTrument[:SELect]`

:CONFigure:ADEMod:PM (No Query Form)

Sets up the analyzer to the default settings for PM signal analysis.
Running this command is equivalent to pressing the following front panel keys:

DEMOD key → **Analog Demod** side key → **PRESET** key → **MEASURE** key → **PM Demod** side key

Syntax `:CONFigure:ADEMod:PM`

Arguments None

Measurement Modes `DEMADEM`

Examples `:CONFigure:ADEMod:PM`
sets up the analyzer to the default settings for PM signal analysis.

Related Commands `:INSTrument[:SELect]`

:CONFigure:ADEMod:PSPectrum (No Query Form)

Sets the analyzer to the default settings for the pulse spectrum measurement.
Running this command is equivalent to pressing the following front panel keys:

DEMOD key → **Analog Demod** side key → **PRESET** key → **Pulse Spectrum** side key

Syntax `:CONFigure:ADEMod:PSPectrum`

Arguments None

Measurement Modes `DEMADEM`

Examples `:CONFigure:ADEMod:PSPectrum`
sets the analyzer to the default settings for the pulse spectrum measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:CCDF (No Query Form)

Sets up the analyzer to the default settings for CCDF measurement. Running this command is equivalent to pressing the following front panel keys:

TIME key → **CCDF** side key → **PRESET** key → **CCDF** side key

Syntax `:CONFigure:CCDF`

Arguments None

Measurement Modes `TIMCCDF`

Examples `:CONFigure:CCDF`
sets up the analyzer to the default settings for CCDF measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:OVIew (No Query Form)

Turns the measurement off in the Demod (modulation analysis) and the Time (time analysis) modes to obtain data in the overview with the :FETCh:OVIew? or the :READ:OVIew? command. Running this command is equivalent to pressing the following front panel keys:

MEASURE key → **Measurement Off** side key

Syntax `:CONFigure:OVIew`

Arguments None

Measurement Modes `DEMADEM`, `TIMCCDF`, `TIMTRAN`

Examples `:CONFigure:OVIew`
turns the measurement off in the Demod and the Time modes.

Related Commands `:FETCh:OVIew?`, `:READ:OVIew?`, `:INSTrument[:SELect]`

:CONFigure:PULSe (No Query Form)

Sets the analyzer to the default settings for pulse characteristics measurement.

TIME key → **Pulse Measurements** side key → **PRESET** key

Syntax `:CONFigure:PULSe`

Arguments None

Measurement Modes `TIMPULSE`

Examples `:CONFigure:PULSe`
sets the analyzer to the default settings for pulse characteristics measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum (No Query Form)

Sets up the analyzer to the default settings for spectrum measurement.
Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** }
side key → **PRESET** key → **Measurement Off** side key

Syntax `:CONFigure:SPECtrum`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CONFigure:SPECtrum`
sets up the analyzer to the default settings for spectrum measurement.,

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:ACPower (No Query Form)

Sets up the analyzer to the default settings for adjacent channel leakage power ratio (ACPR) measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **ACPR** side key

Syntax `:CONFigure:SPECtrum:ACPower`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CONFigure:SPECtrum:ACPower`
sets up the analyzer to the default settings for ACPR measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:CFRequency (No Query Form)

Sets up the analyzer to the default settings for carrier frequency measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **Carrier Frequency** side key

Syntax `:CONFigure:SPECtrum:CFRequency`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CONFigure:SPECtrum:CFRequency`
sets up the analyzer to the default settings for carrier frequency measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:CHPower (No Query Form)

Sets up the analyzer to the default settings for channel power measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **Channel Power** side key

Syntax `:CONFigure:SPECtrum:CHPower`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CONFigure:SPECtrum:CHPower`
sets up the analyzer to the default settings for channel power measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:CNRatio (No Query Form)

Sets up the analyzer to the default settings for carrier-to-noise ratio (C/N) measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **C/N** side key

Syntax `:CONFigure:SPECtrum:CNRatio`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CONFigure:SPECtrum:CNRatio`
sets up the analyzer to the default settings for C/N measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:EBWidth (No Query Form)

Sets up the analyzer to the default settings for emission bandwidth (EBW) measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **EBW** side key

Syntax

```
:CONFigure:SPECtrum:EBWidth
```

Arguments None

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples

```
:CONFigure:SPECtrum:EBWidth
```

sets up the analyzer to the default settings for EBW measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:OBWidth (No Query Form)

Sets up the analyzer to the default settings for occupied bandwidth (OBW) measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **OBW** side key

Syntax

```
:CONFigure:SPECtrum:OBWidth
```

Arguments None

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples

```
:CONFigure:SPECtrum:OBWidth
```

sets up the analyzer to the default settings for OBW measurement:

Related Commands `:INSTrument[:SELect]`

:CONFigure:SPECtrum:SPURious (No Query Form)

The following example sets up the analyzer to the default settings for spurious emission measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → { **Spectrum Analyzer** | **S/A with Spectrogram** | **Real Time S/A** } side key → **PRESET** key → **MEASURE** key → **Spurious** side key

Syntax `:CONFigure:SPECtrum:SPURious`

Arguments None

Measurement Modes `SANORMAL`, `SASGRAM`, `SARTIME`

Examples `:CONFigure:SPECtrum:SPURious`
sets up the analyzer to the default settings for spurious signal measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:TFRequency:RTIMe (No Query Form)

Sets up the analyzer to the default settings for the real-time spectrum measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → **Real Time S/A** side key → **PRESET** key

Syntax `:CONFigure:TFRequency:RTIMe`

Arguments None

Measurement Modes `SARTIME`

Examples `:CONFigure:TFRequency:RTIMe`
sets up the analyzer to the default settings for the real-time spectrum measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:TFRequency:SGRam (No Query Form)

Sets up the analyzer to the default settings for the spectrogram measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → **S/A with Spectrogram** side key → **PRESET** key

Syntax `:CONFigure:TFRequency:SGRam`

Arguments None

Measurement Modes `SASGRAM`

Examples `:CONFigure:TFRequency:SGRam`
sets up the analyzer to the default settings for the spectrogram measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:TRANsient:FVTime (No Query Form)

Sets up the analyzer to the default settings for frequency vs. time measurement. Running this command is equivalent to pressing the following front panel keys:

TIME key → **Transient** side key → **PRESET** key → **MEASURE** key → **Frequency vs. Time** side key

Syntax `:CONFigure:TRANsient:FVTime`

Arguments None

Measurement Modes `TIMTRAN`

Examples `:CONFigure:TRANsient:FVTime`
sets up the analyzer to the default settings for frequency vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:TRANsient:IQVTime (No Query Form)

Sets up the analyzer to the default settings for IQ level vs. time measurement. Running this command is equivalent to pressing the following front panel keys:

TIME key → **Transient** side key → **PRESET** key → **MEASURE** key → **IQ vs. Time** side key

Syntax `:CONFigure:TRANsient:IQVTime`

Arguments None

Measurement Modes `TIMTRAN`

Examples `:CONFigure:TRANsient:IQVTime`
sets up the analyzer to the default settings for IQ level vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:CONFigure:TRANsient:PVTime (No Query Form)

Sets up the analyzer to the default settings for power vs. time measurement. Running this command is equivalent to pressing the following front panel keys:

S/A key → **Transient** side key → **PRESET** key → **MEASURE** key → **Power vs. Time** side key

Syntax `:CONFigure:TRANsient:PVTime`

Arguments None

Measurement Modes `TIMTRAN`

Examples `:CONFigure:TRANsient:PVTime`
sets up the analyzer to the default settings for power vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:DISPlay Commands

The :DISPlay commands control how to show measurement data on the screen. These commands are divided into the following subgroups:

Table 2–30: :DISPlay command subgroups

Command header	Function	Refer to:
:DISPlay:CCDF	Control display of the CCDF analysis.	page 2–80
:DISPlay:OVIew	Control the Demod and Time mode overview.	page 2–86
:DISPlay:PULSe:MVIew|:SVIew	Control the main/sub view in the pulse characteristics analysis.	page 2–98
:DISPlay:PULSe:SPECtrum	Control the spectrum view in the pulse characteristics analysis.	page 2–108
:DISPlay:PULSe:WAVeform	Control the time domain view in the pulse characteristics analysis.	page 2–113
:DISPlay:SPECtrum	Control the spectrum view.	page 2–117
:DISPlay:TFRequency	Control the three-dimensional (spectrogram) view.	page 2–127
:DISPlay[:VIEW]	Set the display brightness and format.	page 2–136
:DISPlay:WAVeform	Control time domain view.	page 2–139

***NOTE**. The :DISPlay commands change the measurement display only, and do not affect the analyzer hardware settings.*

Note on Horizontal Scaling

You can expand an acquired waveform vertically and horizontally on screen (but not contract). Use the :DISPlay commands containing :X[:SCALe] or :Y[:SCALe] node to set the expansion range. Refer to each command description for the setting range. Additionally, meet the following requirements for setting the horizontal scale.

The horizontal display range set by the :DISPlay commands must be within the data acquisition range set by the :SENSe commands (see Figure 2–6):

$X_{START} \leq X_{MIN} < X_{STOP}$
$X_{MAX} \leq X_{STOP}$

Where

X_{START}: the beginning of data acquisition range
X_{STOP}: the end of data acquisition range
X_{MIN}: the beginning of data expansion range
X_{MAX}: the end of data expansion range

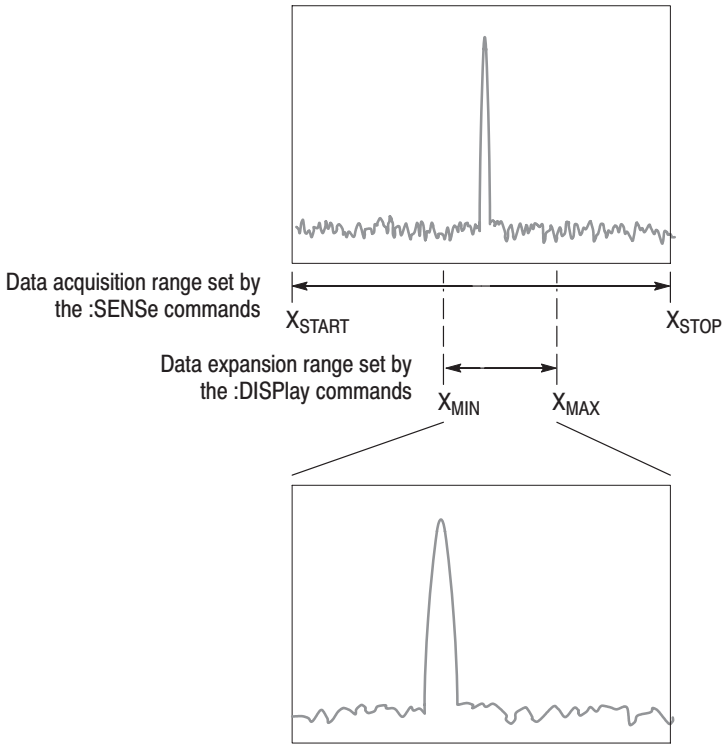


Figure 2–6: Horizontal scale setting requirements

The :DISPlay commands containing the :X[:SCALe] node must meet the above requirements. Figure 2–7 shows an example of the spectrum view. The horizontal scale setting requirements are:

CENTer – SPAN/2 ≤ OFFSet < CENTer + SPAN/2
OFFSet + 10*PDIV ≤ CENTer + SPAN/2

Where

CENTer: [:SENSe]:FREQuency:CENTer value
SPAN: [:SENSe]:FREQuency:SPAN value
OFFSet: :DISPlay:SPECtrum:X[:SCALe]:OFFSet value
PDIVision: :DISPlay:SPECtrum:X[:SCALe]:PDIVision value

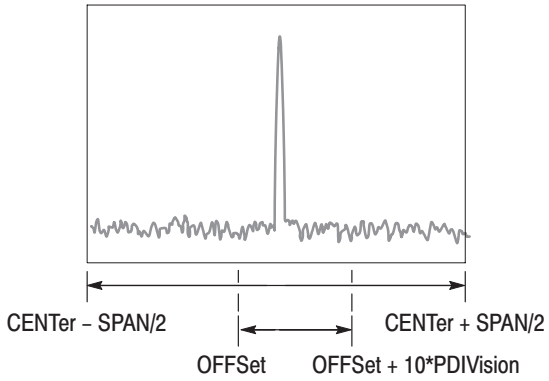


Figure 2–7: Horizontal scale setting requirements for spectrum view

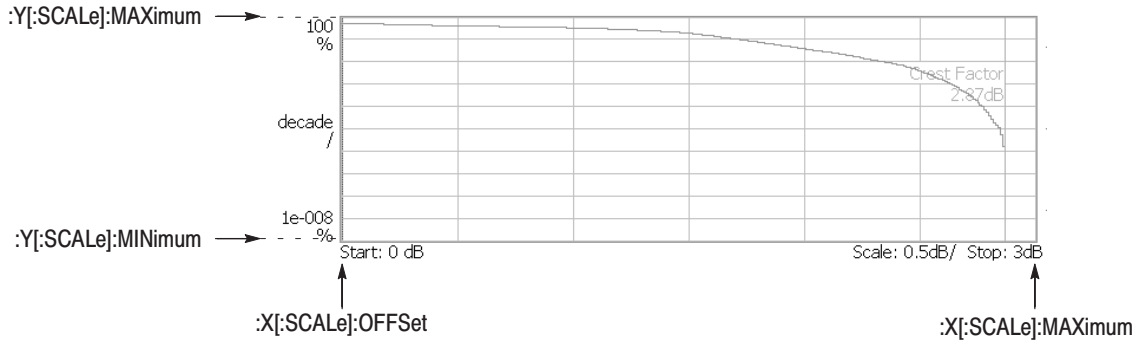
:DISPlay:CCDF Subgroup

The :DISPlay:CCDF commands control the CCDF view.

***NOTE**. To use a command of this group, you must have selected TIMCCDF in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
:DISPlay	
:CCDF	
:LINE	
:GAUSsian	
[:STATe]	<boolean>
:REFerence	
[:STATe]	<boolean>
:STORe	
:X	
[:SCALe]	
:AUTO	<boolean>
:MAXimum	<relative_amplitude>
:OFFSet	<relative_amplitude>
:Y	
[:SCALe]	
:FIT	
:FULL	
:MAXimum	<percent>
:MINimum	<percent>



NOTE: Command header :DISPlay:CCDF is omitted here.

Figure 2–8: :DISPlay:CCDF command setting

:DISPlay:CCDF:LINE:GAUSsian[:STATe](?)

Determines whether to show the Gaussian line in the CCDF view.

Syntax

```
:DISPlay:CCDF:LINE:GAUSsian[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:CCDF:LINE:GAUSsian[:STATe]?
```

Arguments

OFF or 0 hides the Gaussian line.

ON or 1 shows the Gaussian line in the CCDF view.

Measurement Modes

TIMCCDF

Examples

```
:DISPlay:CCDF:LINE:GAUSsian:STATe ON
```
shows the Gaussian line in the CCDF view.

:DISPlay:CCDF:LINE:REFerence[:STATe](?)

Selects whether to show the reference line in the CCDF view. The reference line is stored with the :DISPlay:CCDF:LINE:REFerence:STORe command.

Syntax

```
:DISPlay:CCDF:LINE:REFerence[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:CCDF:LINE:REFerence[:STATe]?
```

Arguments

OFF or 0 hides the reference line.

ON or 1 shows the reference line in the CCDF view.

Measurement Modes

TIMCCDF

Examples

```
:DISPlay:CCDF:LINE:REFerence:STATe ON
```
shows the reference line in the CCDF view.

Related Commands

```
:DISPlay:CCDF:LINE:REFerence:STORe
```

:DISPlay:CCDF:LINE:REFerence:STORe (No Query Form)

Stores the current CCDF trace as a new reference line and automatically enables the reference line display.

Syntax `:DISPlay:CCDF:LINE:REFerence:STORe`

Arguments None

Measurement Modes `TIMCCDF`

Examples `:DISPlay:CCDF:LINE:REFerence:STORe`
stores the current CCDF trace as a new reference line.

Related Commands `:DISPlay:CCDF:LINE:REFerence[:STATe]`

:DISPlay:CCDF:X[:SCALe]:AUTO(?)

Determines whether to automatically set the horizontal, or power, scale in the CCDF view.

Syntax `:DISPlay:CCDF:X[:SCALe]:AUTO { OFF | ON | 0 | 1 }`

`:DISPlay:CCDF:X[:SCALe]:AUTO?`

Arguments `OFF` or `0` specifies that the horizontal scale is set manually (default). Use the :DISPlay:CCDF:X[:SCALe]:MAXimum and the :DISPlay:CCDF:X[:SCALe]-:OFFSet commands, detailed below, to set the horizontal axis.

`ON` or `1` specifies that the horizontal scale is set automatically.

Measurement Modes `TIMCCDF`

Examples `:DISPlay:CCDF:X:SCALe:AUTO ON`
specifies that the horizontal scale is set automatically.

Related Commands `:DISPlay:CCDF:X[:SCALe]:MAXimum`, `:DISPlay:CCDF:X[:SCALe]:OFFSet`

:DISPlay:CCDF:X[:SCALe]:MAXimum(?)

Sets or queries the maximum horizontal, or power, value (right end) in the CCDF view.

Syntax

```
:DISPlay:CCDF:X[:SCALe]:MAXimum <rel_ampl>
:DISPlay:CCDF:X[:SCALe]:MAXimum?
```

Arguments

<rel_ampl>::=<NRf> specifies the maximum horizontal value.
Range: 1 to 100 dB

Measurement Modes

TIMCCDF

Examples

```
:DISPlay:CCDF:X:SCALe:MAXimum 15
```
sets the maximum horizontal value to 15 dB.

Related Commands

:DISPlay:CCDF:X[:SCALe]:AUTO

:DISPlay:CCDF:X[:SCALe]:OFFSet(?)

Sets or queries the start value of the horizontal axis in the CCDF view.

Syntax

```
:DISPlay:CCDF:X[:SCALe]:OFFSet <rel_ampl>
:DISPlay:CCDF:X[:SCALe]:OFFSet?
```

Arguments

<rel_ampl>::=<NRf> specifies the start value of the horizontal axis.
Range: 1 to 100 dB

Measurement Modes

TIMCCDF

Examples

```
:DISPlay:CCDF:X:SCALe:OFFSet 5
```
sets the start value of the horizontal axis to 5 dB.

Related Commands

:DISPlay:CCDF:X[:SCALe]:AUTO

:DISPlay:CCDF:Y[:SCALe]:FIT (No Query Form)

Runs auto-scale on the CCDF view. The auto-scale automatically sets the start value and scale of the vertical axis to display the whole waveform.

Syntax `:DISPlay:CCDF:Y[:SCALe]:FIT`

Arguments None

Measurement Modes `TIMCCDF`

Examples `:DISPlay:CCDF:Y:SCALe:FIT`
runs auto-scale on the CCDF view.

:DISPlay:CCDF:Y[:SCALe]:FULL (No Query Form)

Sets the vertical axis to the default full-scale in the CCDF view.

Syntax `:DISPlay:CCDF:Y[:SCALe]:FULL`

Arguments None

Measurement Modes `TIMCCDF`

Examples `:DISPlay:CCDF:Y:SCALe:FULL`
sets the vertical axis to the default full-scale in the CCDF view.

:DISPlay:CCDF:Y[:SCALe]:MAXimum(?)

Sets or queries the maximum vertical value (top end) in the CCDF view.

Syntax

```
:DISPlay:CCDF:Y[:SCALe]:MAXimum <value>
:DISPlay:CCDF:Y[:SCALe]:MAXimum?
```

Arguments

<value>::=<NRf> sets the maximum vertical value. Range: 10^{-9} to 100%.

Measurement Modes

TIMCCDF

Examples

```
:DISPlay:CCDF:Y:SCALe:MAXimum 80
```

sets the maximum vertical value to 80%.

:DISPlay:CCDF:Y[:SCALe]:MINimum(?)

Sets or queries the minimum vertical value (bottom end) in the CCDF view.

Syntax

```
:DISPlay:CCDF:Y[:SCALe]:MINimum <value>
:DISPlay:CCDF:Y[:SCALe]:MINimum?
```

Arguments

<value>::=<NRf> sets the minimum vertical value. Range: 10^{-9} to 100%.

Measurement Modes

TIMCCDF

Examples

```
:DISPlay:CCDF:Y:SCALe:MINimum 20
```

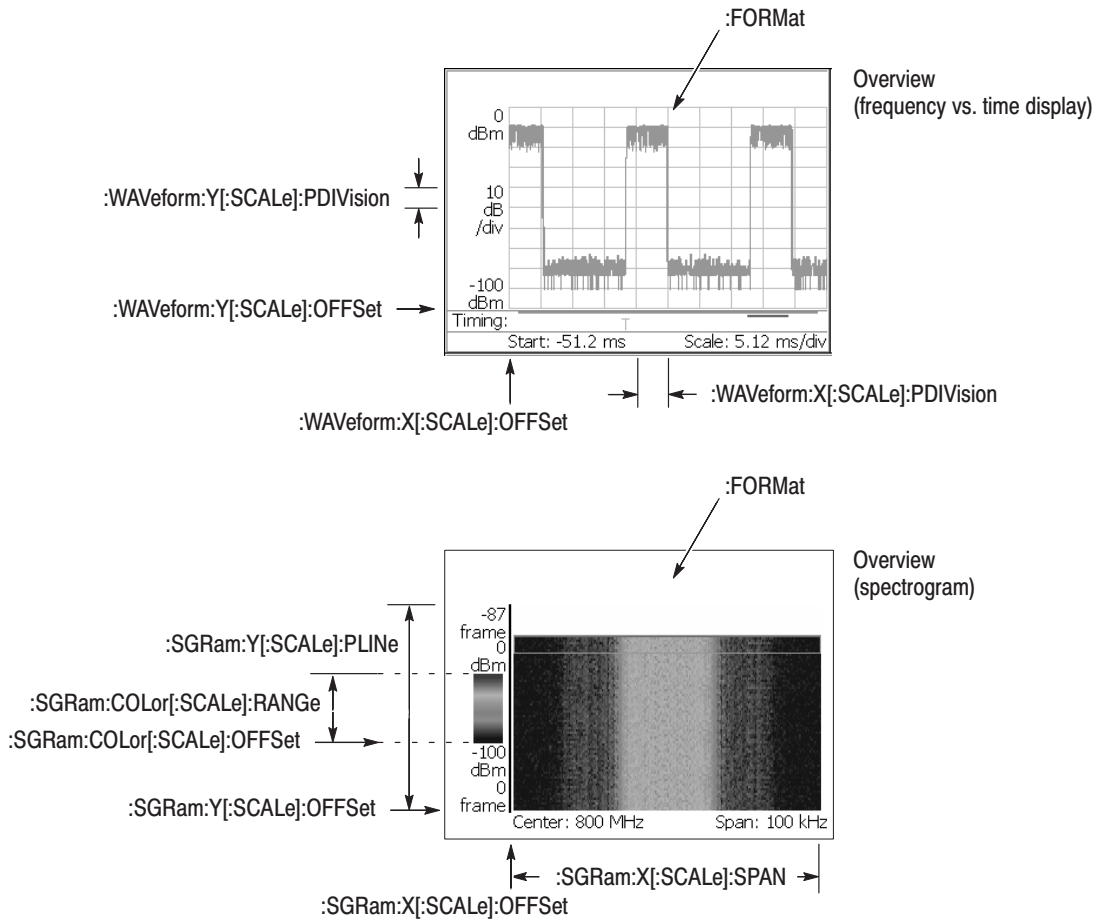
sets the minimum vertical value to 20%.

:DISPlay:OVIew Subgroup

The :DISPlay:OVIew commands set up the overview in the Demod (modulation analysis) and Time (time analyses) modes.

Command Tree

Header	Parameter
:DISPlay	
:OVIew	
:FORMat	WAVeform | SGRam
:SGRam	
:COLor	
[:SCALe]	
:OFFSet	<amplitude>
:RANge	<relative_amplitude>
:X	
[:SCALe]	
:OFFSet	<frequency>
:SPAN	<frequency>
:Y	
[:SCALe]	
:OFFSet	<frame_count>
:PLINe	<frame_count>
:WAVeform	
:X	
[:SCALe]	
:OFFSet	<time>
:PDIVsion	<time>
:Y	
[:SCALe]	
:FIT	
:FULL	
:OFFSet	<amplitude>
:PDIVsion	<amplitude>
:ZOOM	
:COLor	
[:SCALe]	
:OFFSet	<amplitude>
:RANge	<relative_amplitude>
:X	
[:SCALe]	
:OFFSet	<frequency>
:SPAN	<frequency>
:Y	
[:SCALe]	
:OFFSet	<frame_count>
:PLINe	<frame_count>



NOTE: Command header :DISPlay:OVIew is omitted here.

Figure 2–9: :DISPlay:OVIew command setting

:DISPlay:OVIew:FORMat(?)

Selects or queries the overview display format.

Syntax

```
:DISPlay:OVIew:FORMat { WAVeform | SGRam }
:DISPlay:OVIew:FORMat?
```

Arguments

`WAVeform` displays the amplitude vs. time view.

`SGRam` displays the spectrogram.

Measurement Modes

`DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples

```
:DISPlay:OVIew:FORMat SGRam
```
displays the spectrogram view in the overview.

:DISPlay:OVIew:SGRam:COLor[:SCALe]:OFFSet(?)

Sets or queries the minimum value (bottom end) of the color, or amplitude, axis when the overview displays a spectrogram.

Syntax

```
:DISPlay:OVIew:SGRam:COLor[:SCALe]:OFFSet <ampl>
:DISPlay:OVIew:SGRam:COLor[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> specifies the minimum color-axis value in the overview. Range: –200 to +100 dBm.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:SGRam:COLor:SCALe:OFFSet -100
```

sets the minimum color-axis value to –100 dBm.

:DISPlay:OVIew:SGRam:COLor[:SCALe]:RANGe(?)

Sets or queries full-scale of the color, or amplitude, axis when the overview displays a spectrogram.

Syntax

```
:DISPlay:OVIew:SGRam:COLor[:SCALe]:RANGe <rel_ampl>
:DISPlay:OVIew:SGRam:COLor[:SCALe]:RANGe?
```

Arguments

<rel_ampl>::={ 10 | 20 | 50 | 100 } [dB] specifies the full-scale of the color axis.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:SGRam:COLor:SCALe:RANGe 100
```

sets full-scale of the color axis to 100 dB.

:DISPlay:OVIew:SGRam:X[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or frequency, value (left end) when the overview displays a spectrogram.

Syntax

```
:DISPlay:OVIew:SGRam:X[:SCALe]:OFFSet <freq>
:DISPlay:OVIew:SGRam:X[:SCALe]:OFFSet?
```

Arguments

<freq>::=<NRf> specifies the minimum horizontal value of the spectrogram. The valid range depends on the setting in the [:SENSe]:FREQuency:BAND command. Refer to Table 2–39 on page 2–271.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:SGRam:X:SCALe:OFFSet 100MHz
```

sets the minimum horizontal value to 100 MHz.

:DISPlay:OVIew:SGRam:X[:SCALe]:SPAN(?)

Sets or queries the span of the horizontal, or frequency, axis when the overview displays a spectrogram.

Syntax

```
:DISPlay:OVIew:SGRam:X[:SCALe]:SPAN <freq>
:DISPlay:OVIew:SGRam:X[:SCALe]:SPAN?
```

Arguments

<freq>::=<NRf> specifies the horizontal span.
Range: 100 Hz to 10 MHz (RF)
100 Hz to 20 MHz (baseband with option 05)

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:SGRam:X:SCALe:SPAN 100kHz
```

sets the span to 100 kHz.

:DISPlay:OVIew:SGRam:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum vertical, or frame number, value (bottom end) when the overview displays a spectrogram.

Syntax

```
:DISPlay:OVIew:SGRam:Y[:SCALe]:OFFSet <value>
:DISPlay:OVIew:SGRam:Y[:SCALe]:OFFSet?
```

Arguments

<value>::=<NR1> specifies the minimum vertical value of the spectrogram. Range: Frame # –63999 to 0.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:SGRam:Y:SCALe:OFFSet -100
```

sets the minimum vertical value to frame # –100.

:DISPlay:OVIew:SGRam:Y[:SCALe]:PLINe(?)

Sets or queries the vertical scale (the number of frames per line) when the overview displays a spectrogram.

Frames are thinned out from all the acquired framed data at intervals of the number of frames specified in this command, before the spectrogram is displayed. For example, if you set the argument to 5, the data will be displayed every 5 frames.

Syntax

```
:DISPlay:OVIew:SGRam:Y[:SCALe]:PLINe <value>
:DISPlay:OVIew:SGRam:Y[:SCALe]:PLINe?
```

Arguments

<value>::=<NR1> specifies the vertical scale for the spectrogram. Range: 1 to 1024 frames per line.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:SGRam:Y:SCALe:PLINe 5
```

displays the data in the spectrogram every 5 frames.

:DISPlay:OVIew:WAVeform:X[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or time, value (left end) when the overview displays an amplitude vs. time waveform.

Syntax

```
:DISPlay:OVIew:WAVeform:X[:SCALe]:OFFSet <time>
:DISPlay:OVIew:WAVeform:X[:SCALe]:OFFSet?
```

Arguments

<time>::=<NRf> specifies the minimum horizontal value. Range: –32000 to 0 s.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:WAVeform:X:SCALe:OFFSet -100us
```
sets the minimum horizontal value to –100 μs.

:DISPlay:OVIew:WAVeform:X[:SCALe]:PDIVision(?)

Sets or queries the horizontal, or time, scale (per division) when the overview displays an amplitude vs. time view.

Syntax

```
:DISPlay:OVIew:WAVeform:X[:SCALe]:PDIVision <time>
:DISPlay:OVIew:WAVeform:X[:SCALe]:PDIVision?
```

Arguments

<time>::=<NRf> specifies the horizontal scale. Range: 0 to 3200 s/div.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:WAVeform:X:SCALe:PDIVision 10.0E-6
```
sets the horizontal scale to 10 μs/div.

:DISPlay:OVIew:WAVeform:Y[:SCALe]:FIT (No Query Form)

Runs the auto-scale on the overview. The auto-scale automatically sets the start value and scale of the vertical axis to display the whole waveform.

Syntax `:DISPlay:OVIew:WAVeform:Y[:SCALe]:FIT`

Arguments None

Measurement Modes `DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples `:DISPlay:OVIew:WAVeform:Y:SCALe:FIT`
runs the auto-scale on the overview.

:DISPlay:OVIew:WAVeform:Y[:SCALe]:FULL (No Query Form)

Sets the vertical axis in the overview to the default full-scale.

Syntax `:DISPlay:OVIew:WAVeform:Y[:SCALe]:FULL`

Arguments None

Measurement Modes `DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples `:DISPlay:OVIew:WAVeform:Y:SCALe:FULL`
sets the overview's vertical axis to the default full-scale.

:DISPlay:OVIew:WAVeform:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum vertical, or amplitude, value (bottom end) when the overview displays an amplitude vs. time waveform.

Syntax

```
:DISPlay:OVIew:WAVeform:Y[:SCALe]:OFFSet <ampl>
:DISPlay:OVIew:WAVeform:Y[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> specifies the minimum vertical value.
Range: –200 to 0 dBm.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:WAVeform:Y:SCALe:OFFSet -100
```
sets the minimum vertical value to –100 dBm.

:DISPlay:OVIew:WAVeform:Y[:SCALe]:PDIVision(?)

Sets or queries the vertical, or amplitude, scale (per division) when the overview displays an amplitude vs. time waveform.

Syntax

```
:DISPlay:OVIew:WAVeform:Y[:SCALe]:PDIVision <ampl>
:DISPlay:OVIew:WAVeform:Y[:SCALe]:PDIVision?
```

Arguments

<ampl>::=<NRf> specifies the vertical scale. Range: 0 to 30 dB/div.

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:DISPlay:OVIew:WAVeform:Y:SCALe:PDIVision 10
```
sets the vertical scale to 10 dB/div.

:DISPlay:OVIew:ZOOM:COLor[:SCALe]:OFFSet(?)

Sets or queries the minimum value (bottom) of the color, or amplitude, axis of the spectrogram with zoom function.

Syntax

```
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:OFFSet <ampl>
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> specifies the minimum color-axis value of the spectrogram with zoom function. Range: –200 to +100 dBm.

Measurement Modes

SAZRTIME

Examples

```
:DISPlay:OVIew:ZOOM:COLor:SCALe:OFFSet -100
```

sets the minimum color-axis value to –100 dBm.

:DISPlay:OVIew:ZOOM:COLor[:SCALe]:RANGe(?)

Sets or queries full-scale value of the color, or amplitude, axis of the spectrogram with zoom function.

Syntax

```
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:RANGe <rel_ampl>
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:RANGe?
```

Arguments

<rel_ampl>::={ 10 | 20 | 50 | 100 } [dB] specifies the full-scale value of the color axis of the spectrogram with zoom function.

Measurement Modes

SAZRTIME

Examples

```
:DISPlay:OVIew:ZOOM:COLor:SCALe:RANGe 100
```

sets full-scale value of the color axis to 100 dB.

:DISPlay:OVIew:ZOOM:X[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or frequency, value (left edge) of the spectrogram with zoom function.

Syntax

```
:DISPlay:OVIew:ZOOM:X[:SCALe]:OFFSet <freq>
:DISPlay:OVIew:ZOOM:X[:SCALe]:OFFSet?
```

Arguments

<freq>::=<NRf> specifies the minimum horizontal value of the spectrogram with zoom function. Refer to *Note on Horizontal Scaling* on page 2–78 for setting the scale.

Measurement Modes

SAZRTIME

Examples

```
:DISPlay:OVIew:ZOOM:X:SCALe:OFFSet 100MHz
```

sets the minimum horizontal value to 100 MHz.

:DISPlay:OVIew:ZOOM:X[:SCALe]:SPAN(?)

Sets or queries the span of the horizontal, or frequency, axis of the spectrogram with zoom function.

Syntax

```
:DISPlay:OVIew:ZOOM:X[:SCALe]:SPAN <freq>
:DISPlay:OVIew:ZOOM:X[:SCALe]:SPAN?
```

Arguments

<freq>::=<NRf> specifies the horizontal span of the spectrogram with zoom function. Refer to *Note on Horizontal Scaling* on page 2–78 for setting the scale.

Measurement Modes

SAZRTIME

Examples

```
:DISPlay:OVIew:ZOOM:X:SCALe:SPAN 100kHz
```

sets the span to 100 kHz.

:DISPlay:OVIew:ZOOM:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum vertical, or frame number, value (bottom) of the spectrogram with zoom function.

Syntax

```
:DISPlay:OVIew:ZOOM:Y[:SCALe]:OFFSet <value>
:DISPlay:OVIew:ZOOM:Y[:SCALe]:OFFSet?
```

Arguments

<value>::=<NR1> specifies the minimum vertical value of the spectrogram with zoom function. Range: Frame # –63999 to 0.

Measurement Modes

SAZRTIME

Examples

```
:DISPlay:OVIew:ZOOM:Y:SCALe:OFFSet -100
```

sets the minimum vertical value to frame # –100.

:DISPlay:OVIew:ZOOM:Y[:SCALe]:PLINe(?)

Sets or queries the vertical scale (the number of frames per line) of the spectrogram with zoom function.

Frames are thinned out from all the acquired framed data at intervals of the number of frames specified in this command, before the spectrogram is displayed. For example, if you set the argument to 5, the data will be displayed every 5 frames.

Syntax

```
:DISPlay:OVIew:ZOOM:Y[:SCALe]:PLINe <value>
:DISPlay:OVIew:ZOOM:Y[:SCALe]:PLINe?
```

Arguments

<value>::=<NR1> specifies the vertical scale for the spectrogram with zoom function. Range: 1 to 1024 frames per line.

Measurement Modes

SAZRTIME

Examples

```
:DISPlay:OVIew:ZOOM:Y:SCALe:PLINe 5
```

displays the data in the spectrogram every 5 frames.

:DISPlay:PULSe:MVIew|:SVIew Subgroup

The :DISPlay:PULSe:MVIew|:SVIew commands control display of the main view (pulse result table) and subview in the pulse charactristics analysis.

***NOTE**. To use a command from this group, you must have selected TIMPULSE (pulse characteristics analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
:DISPlay	
:PULSe	
:MVIew	
:RESult	
:CHPower	<boolean>
:DCYCle	<boolean>
:EBWidth	<boolean>
:FREQuency	<boolean>
:OBWidth	<boolean>
:OORatio	<boolean>
:PERiod	<boolean>
:PHASe	<boolean>
:PPOWer	<boolean>
:RIPPle	<boolean>
:WIDTh	<boolean>
:SVIew	
:FORMat	WIDTh | PPOWer | OORatio | RIPPle | PERIiod | DCYCle | PHASe | CHPower | OBWidth | EBWidth | FREQuency
:GUIDelines	<boolean>
:RANGe	ADAPtive | MAXimum
:RESult	SINGle | ALL
:SELect	<numeric_value>

:DISPlay:PULSe:MVIew:RESult:CHPower(?)

Determines whether to show channel power measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:CHPower { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:CHPower?
```

Arguments

OFF or 0 hides channel power measurement results in the pulse result table.

ON or 1 shows channel power measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:CHPower ON
```
shows channel power measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:DCYCle(?)

Determines whether to show duty cycle measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:DCYCle { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:DCYCle?
```

Arguments

OFF or 0 hides duty cycle measurement results in the pulse result table.

ON or 1 shows duty cycle measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:DCYCle ON
```
shows duty cycle measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:EBWidth(?)

Determines whether to show EBW (Emission Bandwidth) measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:EBWidth { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:EBWidth?
```

Arguments

OFF or 0 hides EBW measurement results in the pulse result table.

ON or 1 shows EBW measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:EBWidth ON
```
shows EBW measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:FREQuency(?)

Determines whether to show frequency deviation measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:FREQuency { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:FREQuency?
```

Arguments

OFF or 0 hides frequency deviation measurement results in the pulse result table.

ON or 1 shows frequency deviation measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:FREQuency ON
```
shows frequency deviation measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:OBWidth(?)

Determines whether to show OBW (Occupied Bandwidth) measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:OBWidth { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:OBWidth?
```

Arguments

OFF or 0 hides OBW measurement results in the pulse result table.

ON or 1 shows OBW measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:OBWidth ON
```
shows OBW measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:OORatio(?)

Determines whether to show on/off-ratio measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:OORatio { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:OORatio?
```

Arguments

OFF or 0 hides on/off-ratio measurement results in the pulse result table.

ON or 1 shows on/off-ratio measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:OORatio ON
```
shows on/off-ratio measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:PERiod(?)

Determines whether to show pulse repetition interval measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:PERiod { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:PERiod?
```

Arguments

OFF or 0 hides pulse repetition interval measurement results in the pulse result table.

ON or 1 shows pulse repetition interval measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:PERiod ON
```
shows pulse repetition interval measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:PHASe(?)

Determines whether to show pulse-pulse phase measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:PHASe { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:PHASe?
```

Arguments

OFF or 0 hides pulse-pulse phase measurement results in the pulse result table.

ON or 1 shows pulse-pulse phase measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:PHASe ON
```
shows pulse-pulse phase measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:PPOWer(?)

Determines whether to show peak power measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:PPOWer { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:PPOWer?
```

Arguments

OFF or 0 hides peak power measurement results in the pulse result table.

ON or 1 shows peak power measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:PPOWer ON
```
shows peak power measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:RIPPle(?)

Determines whether to show pulse ripple measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:RIPPle { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:RIPPle?
```

Arguments

OFF or 0 hides pulse ripple measurement results in the pulse result table.

ON or 1 shows pulse ripple measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:RIPPle ON
```
shows pulse ripple measurement results in the pulse result table.

:DISPlay:PULSe:MVIew:RESult:WIDTh(?)

Determines whether to show pulse width measurement results in the pulse result table.

Syntax

```
:DISPlay:PULSe:MVIew:RESult:WIDTh { 0 | 1 | OFF | ON }
:DISPlay:PULSe:MVIew:RESult:WIDTh?
```

Arguments

OFF or 0 hides peak power measurement results in the pulse result table.

ON or 1 shows peak power measurement results in the pulse result table.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:MVIew:RESult:WIDTh ON
```
shows peak power measurement results in the pulse result table.

:DISPlay:PULSe:SVIew:FORMat(?)

Selects or queries the display format of the subview in the pulse characteristics analysis.

Syntax

```
:DISPlay:PULSe:SVIew:FORMat { WIDTh | PPOWer | OORatio | RIPPle
| PERIiod | DCYCle | PHASe | CHPower | OBWidth | EBWidth
| FREQuency }

:DISPlay:PULSe:SVIew:FORMat?
```

Arguments

The arguments and display formats are listed below:

Table 2–31: Subview display format

Argument	Display format
WIDTh	Pulse width
PPOWer	Peak power in the pulse-on time
OORatio	Difference between the on-time power and off-time power
RIPPle	Difference between the maximum and minimum power in the pulse-on time
PERiod	Time between a pulse rising edge and the next pulse rising edge
DCYCle	Ratio of the pulse width to the pulse repetition interval (PRI)
PHASe	Phase at a certain point in each pulse
CHPower	Channel power of the pulse-on time spectrum
OBWidth	OBW of the pulse-on time spectrum
EBWidth	EBW of the pulse-on time spectrum
FREQuency	Frequency deviation of the pulse-on time

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SVIew:FORMat WIDTh
```

displays the pulse width measurement result and waveform in the subview.

:DISPlay:PULSe:SVIew:GUIDelines(?)

Determines whether to show the guidelines in the subview.

Syntax

```
:DISPlay:PULSe:SVIew:GUIDelines { 0 | 1 | OFF | ON }
:DISPlay:PULSe:SVIew:GUIDelines?
```

Arguments

OFF or 0 hides the guidelines in the subview.

ON or 1 shows the guidelines in the subview (default).

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SVIew:GUIDelines ON
```
shows the guidelines in the subview.

:DISPlay:PULSe:SVIew:RANGe(?)

Selects or queries how to set the horizontal scale in the subview.

Syntax

```
:DISPlay:PULSe:SVIew:RANGe { ADAPtive | MAXimum }
:DISPlay:PULSe:SVIew:RANGe?
```

Arguments

ADAPtive adjusts the horizontal scale for each pulse to fit the pulse width to the subview (default).

MAXimum adjusts the horizontal scale to fit the maximum pulse width in the analysis range to the subview.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SVIew:RANGe ADAPtive
```
adjusts the horizontal scale for each pulse to fit the pulse width to the subview.

:DISPlay:PULSe:SVIew:RESult(?)

Selects or queries how to show the result graph in the subview.

Syntax

```
:DISPlay:PULSe:SVIew:RESult { SINGle | ALL }
:DISPlay:PULSe:SVIew:RESult?
```

Arguments

SINGle shows the measurement result and waveform for a pulse in the subview. Select the pulse using the :DISPlay:PULSe:SVIew:SELect command.

ALL shows the measurement results for all pulses in the subview, representing pulse numbers along the horizontal axis and measurement values along the vertical axis.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SVIew:RESult SINGle
```
shows the measurement result and waveform for a pulse in the subview.

Related Commands

:DISPlay:PULSe:SVIew:SELect

:DISPlay:PULSe:SVIew:SELect(?)

Selects or queries a pulse to measure when you select SINGle with the :DISPlay :PULSe:SVIew:RESult command.

Syntax

```
:DISPlay:PULSe:SVIew:SELect <number>
:DISPlay:PULSe:SVIew:SELect?
```

Arguments

<number>::=<NR1> specifies the a single pulse number. 0 (zero) represents the latest pulse. The older pulse has the larger negative number. Range: –999 to 0

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SVIew:RESult -125
```
specifies pulse #–125 to display in the subview.

:DISPlay:PULSe:SPECtrum Subgroup

The :DISPlay:PULSe:SPECtrum commands control the spectrum display in the frequency domain measurements under the pulse characteristics analysis.

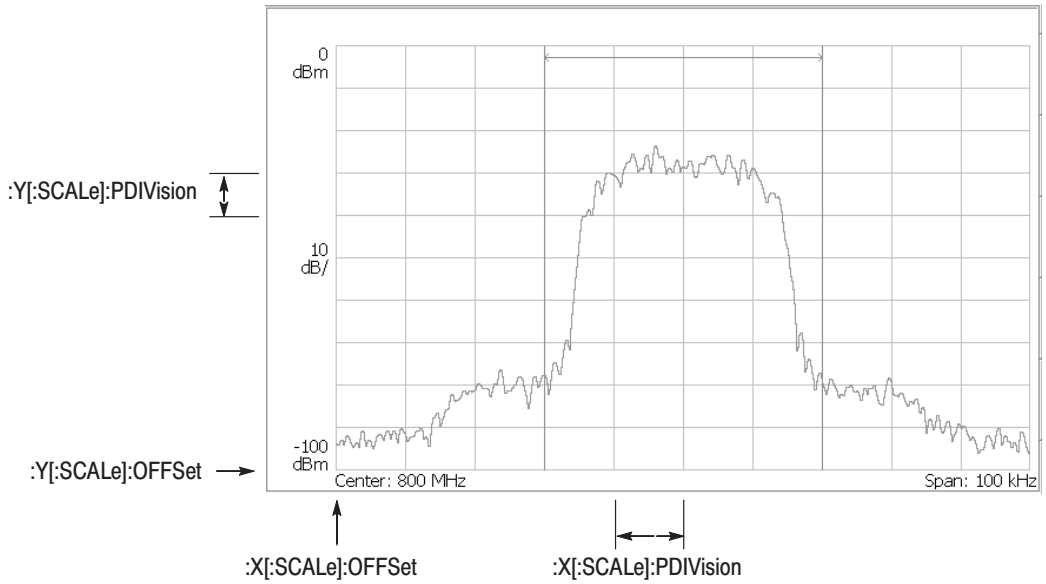
These commands are valid when you select one of the following items using the :DISplay:PULSe:SVIew:FORMat command.

- CHPower (channel power)
- OBWidth (OBW)
- EBWidth (EBW)

***NOTE**. To use a command from this group, you must have selected TIMPULSE (pulse characteristics analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
:DISPlay	
:PULSe	
:SPECtrum	
:X	
[:SCALe]	
:OFFSet	<numeric_value>
:PDIVision	<numeric_value>
:Y	
[:SCALe]	
:FIT	
:FULL	
:OFFSet	<numeric_value>
:PDIVision	<numeric_value>



NOTE: Command header :DISPlay:PULSe:SPECtrum is omitted here.

Figure 2–10: :DISPlay:PULSe:SPECtrum command setting

:DISPlay:PULSe:SPECtrum:X[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or frequency, value (left edge) in the spectrum view.

Syntax

```
:DISPlay:PULSe:SPECtrum::X[:SCALe]:OFFSet <freq>
:DISPlay:PULSe:SPECtrum::X[:SCALe]:OFFSet?
```

Arguments

<freq>::=<NRf> specifies the minimum horizontal value in the spectrum view. For the setting range, refer to *Note on Horizontal Scaling* on page 2–78.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SPECtrum:X:SCALe:OFFSet 100MHz
```

sets the minimum horizontal value to 100 MHz.

:DISPlay:PULSe:SPECtrum:X[:SCALe]:PDIVision(?)

Sets or queries the horizontal, or frequency, scale (per division) in the spectrum view.

Syntax

```
:DISPlay:PULSe:SPECtrum:X[:SCALe]:PDIVision <freq>
:DISPlay:PULSe:SPECtrum:X[:SCALe]:PDIVision?
```

Arguments

<freq>::=<NRf> specifies the horizontal scale.
For the setting range, refer to *Note on Horizontal Scaling* on page 2–78.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SPECtrum:X:SCALe:PDIVision 100.0E+3
```

sets the horizontal scale to 100 kHz/div.

:DISPlay:PULSe:SPECtrum:Y[:SCALe]:FIT (No Query Form)

Runs the auto-scale on the spectrum view. The auto-scale automatically sets the start value and scale of the vertical axis to fit the waveform to the screen.

Syntax `:DISPlay:PULSe:SPECtrum:Y[:SCALe]:FIT`

Arguments None

Measurement Modes `TIMPULSE`

Examples `:DISPlay:PULSe:SPECtrum:Y:SCALe:FIT`
runs the auto-scale on the spectrum view.

:DISPlay:PULSe:SPECtrum:Y[:SCALe]:FULL (No Query Form)

Sets the vertical axis to the default full-scale value in the spectrum view.

Syntax `:DISPlay:PULSe:SPECtrum:Y[:SCALe]:FULL`

Arguments None

Measurement Modes `TIMPULSE`

Examples `:DISPlay:PULSe:SPECtrum:Y:SCALe:FULL`
sets the vertical axis to the default full-scale value in the spectrum view.

:DISPlay:PULSe:SPECtrum:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum vertical, or amplitude, value (bottom) in the spectrum view.

Syntax

```
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:OFFSet <ampl>
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> sets the minimum vertical value. Range: –200 to 0 dBm.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SPECtrum:Y:SCALe:OFFSet -100
```

sets the minimum vertical value to –100 dBm.

:DISPlay:PULSe:SPECtrum:Y[:SCALe]:PDIVision(?)

Sets or queries the vertical, or amplitude, scale (per division) in the spectrum view.

Syntax

```
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:PDIVision <ampl>
:DISPlay:PULSe:SPECtrum:Y[:SCALe]:PDIVision?
```

Arguments

<ampl>::=<NRf> specifies the vertical scale in the spectrum view. Range: 0 to 10 dB/div.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:SPECtrum:Y:SCALe:PDIVision 10
```

sets the vertical scale to 10 dB/div.

:DISPlay:PULSe:WAVeform Subgroup

The :DISPlay:PULSe:WAVeform commands control the time domain display in the time domain measurements under the pulse characteristics analysis.

These commands are valid when you select one of the following items using the :DISplay:PULSe:SVIew:FORMat command.

- WIDTh (pulse width)
- PPOWer (peak power)
- OORatio (pulse on/off ratio)
- RIPPle (pulse ripple)
- PERiod (pulse period)
- DCYCle (duty cycle)
- PHASe (pulse-pulse phase)
- FREQuency (frequency deviation)

***NOTE**. To use a command from this group, you must have selected TIMPULSE (pulse characteristics analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
:DISPlay	
:PULSe	
:WAVeform	
:X	
[:SCALe]	
:OFFSet	<numeric_value>
:PDIVision	<numeric_value>
:Y	
[:SCALe]	
:FIT	
:FULL	
:OFFSet	<numeric_value>
:PDIVision	<numeric_value>

:DISPlay:PULSe:WAVeform:X[:SCALe]:OFFSet(?)

Sets or queries the minimum value of the horizontal axis (left edge) in the time domain display.

Syntax

```
:DISPlay:PULSe:WAVeform:X[:SCALe]:OFFSet <time>
:DISPlay:PULSe:WAVeform:X[:SCALe]:OFFSet?
```

Arguments

<time>::=<NRf> sets the minimum horizontal value. Range: –32000 to 0 s. For the setting range, refer to *Note on Horizontal Scaling* on page 2–78.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:WAVeform:X:SCALe:OFFSet -100us
```

sets the minimum horizontal value to –100 μs.

:DISPlay:PULSe:WAVeform:X[:SCALe]:PDIVision(?)

Sets or queries the horizontal, or time, scale (per division) in the time domain display.

Syntax

```
:DISPlay:PULSe:WAVeform:X[:SCALe]:PDIVision <time>
:DISPlay:PULSe:WAVeform:X[:SCALe]:PDIVision?
```

Arguments

<time>::=<NRf> specifies the horizontal scale. Range: 0 to 3200 s/div. For the setting range, refer to *Note on Horizontal Scaling* on page 2–78.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:WAVeform:X:SCALe:PDIVision 10us
```

sets the horizontal scale to 10 μs/div.

:DISPlay:PULSe:WAVeform:Y[:SCALe]:FIT (No Query Form)

Runs the auto-scale on the time domain display. The auto-scale automatically sets the start value and scale of the vertical axis to fit the waveform to the screen.

Syntax `:DISPlay:PULSe:WAVeform:Y[:SCALe]:FIT`

Arguments None

Measurement Modes `TIMPULSE`

Examples `:DISPlay:PULSe:WAVeform:Y:SCALe:FIT`
runs the auto-scale.

:DISPlay:PULSe:WAVeform:Y[:SCALe]:FULL (No Query Form)

Sets the vertical axis in the time domain display to the default full-scale value.

Syntax `:DISPlay:PULSe:WAVeform:Y[:SCALe]:FULL`

Arguments None

Measurement Modes `TIMPULSE`

Examples `:DISPlay:PULSe:WAVeform:Y:SCALe:FULL`
sets the vertical axis in the time domain display to the default full-scale value.

:DISPlay:PULSe:WAVeform:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum value (bottom) of the vertical axis in the time domain display.

Syntax

```
:DISPlay:PULSe:WAVeform:Y[:SCALe]:OFFSet <ampl>
:DISPlay:PULSe:WAVeform:Y[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> specifies the minimum value of the vertical axis. The valid range depends on the display format. Refer to Table D–1 in *Appendix D*.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:WAVeform:Y:SCALe:OFFSet -100
```

sets the minimum vertical value to –100 dBm.

:DISPlay:PULSe:WAVeform:Y[:SCALe]:PDIVision(?)

Sets the vertical axis scale (per division) in the time domain display.

Syntax

```
:DISPlay:PULSe:WAVeform:Y[:SCALe]:PDIVision <ampl>
:DISPlay:PULSe:WAVeform:Y[:SCALe]:PDIVision?
```

Arguments

<ampl>::=<NRf> specifies the vertical scale. The valid range depends on the display format. Refer to Table D–1 in *Appendix D*.

Measurement Modes

TIMPULSE

Examples

```
:DISPlay:PULSe:WAVeform:Y:SCALe:PDIVision 10
```

sets the vertical scale to 10 dB/div

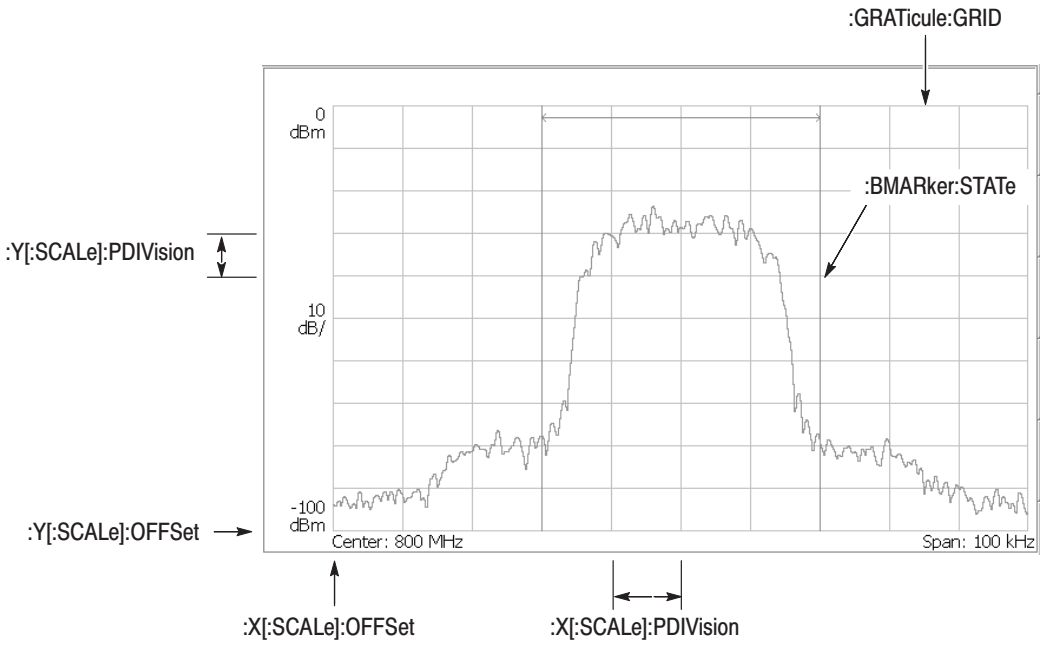
:DISPlay:SPECtrum Subgroup

The :DISPlay:SPECtrum commands control the spectrum view.

***NOTE**. To use a command of this group, you must have a spectrum that is currently displayed in the view, regardless of the measurement mode.*

Command Tree

Header	Parameter
:DISPlay	
:SPECtrum	
:BMARker	
:STATe	<boolean>
:GRATicule	
:GRID	OFF | FIX | FLEX
:MLINe	
:AMPLitude	
:INTerval	<numeric_value>
:OFFSet	<numeric_value>
[:STATe]	<boolean>
:ANNotation	
[:STATe]	<boolean>
:FREQuency	
:INTerval	<numeric_value>
:OFFSet	<numeric_value>
[:STATe]	<boolean>
:X	
[:SCALe]	
:OFFSet	<frequency>
:PDIVsion	<frequency>
:Y	
[:SCALe]	
:FIT	
:FULL	
:OFFSet	<amplitude>
:PDIVsion	<amplitude>



NOTE: Command header :DISPlay:SPECtrum is omitted here.

Figure 2–11: :DISPlay:SPECtrum command setting

:DISPlay:SPECtrum:BMARker:STATe(?)

Determines whether to show the band power marker.

Syntax

```
:DISPlay:SPECtrum:BMARker:STATe { OFF | ON | 0 | 1 }
:DISPlay:SPECtrum:BMARker:STATe?
```

Arguments

OFF or 0 hides the band power marker.

ON or 1 shows the band power marker.

Measurement Modes

All

Examples

```
:DISPlay:SPECtrum:BMARker:STATe ON
```

shows the band power marker.

:DISPlay:SPECtrum:GRATicule:GRID(?)

Selects or queries how the graticule is displayed.

***NOTE**. This command is available in the S/A (spectrum analysis) mode except Real Time S/A.*

Syntax

```
:DISPlay:SPECtrum:GRATicule:GRID { OFF | FIX | FLEX }
:DISPlay:SPECtrum:GRATicule:GRID?
```

Arguments

OFF hides the graticule.

FIX always shows the 10 divisions × 10 divisions graticule.

FLEX shows the graticule so that one division is set in 1-2-5 sequence.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:DISPlay:SPECtrum:GRATicule:GRID FIX
```

always shows the 10 × 10 graticule.

:DISPlay:SPECtrum:MLINe:AMPLitude:INTerval(?)

Sets or queries the interval of the amplitude multi display lines in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:AMPLitude:INTerval <value>
:DISPlay:SPECtrum:MLINe:AMPLitude:INTerval?
```

Arguments

<value>::=<NRf> sets the interval of the amplitude multi display lines. Range: 0 to 100 dB.

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:AMPLitude:INTerval 5
```

sets the interval to 5 dB.

:DISPlay:SPECtrum:MLINe:AMPLitude:OFFSet(?)

Sets or queries the offset of the amplitude multi display lines in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:AMPLitude:OFFSet <value>
:DISPlay:SPECtrum:MLINe:AMPLitude:OFFSet?
```

Arguments

<value>::=<NRf> sets the offset of the amplitude multi display lines. Range: –100 to 0 dBm.

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:AMPLitude:OFFSet -10
```

sets the offset to –10 dBm.

:DISPlay:SPECtrum:MLINe:AMPLitude[:STATe](?)

Determines whether to show the amplitude multi display lines in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:AMPLitude[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:SPECtrum:MLINe:AMPLitude[:STATe]?
```

Arguments

OFF or 0 hides the amplitude multi display lines.

ON or 1 shows the amplitude multi display lines.

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:AMPLitude:STATe ON
```
shows the amplitude multi display lines.

:DISPlay:SPECtrum:MLINe:ANNotation[:STATe](?)

Determines whether to show the multi display lines readout in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:ANNotation[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:SPECtrum:MLINe:ANNotation[:STATe]?
```

Arguments

OFF or 0 hides the multi display lines readout.

ON or 1 shows the multi display lines readout.

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:ANNotation:STATe ON
```
shows the readout.

:DISPlay:SPECtrum:MLINe:FREQuency:INTerval(?)

Sets or queries the interval of the frequency multi display lines in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:FREQuency:INTerval <value>
:DISPlay:SPECtrum:MLINe:FREQuency:INTerval?
```

Arguments

<value>::=<NRf> sets the interval of the frequency multi display lines. Range: 0 to full span (Hz).

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:FREQuency:INTerval 1MHz
```

sets the interval to 1 MHz.

:DISPlay:SPECtrum:MLINe:FREQuency:OFFSet(?)

Sets or queries the offset of the frequency multi display lines in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:FREQuency:OFFSet <value>
:DISPlay:SPECtrum:MLINe:FREQuency:OFFSet?
```

Arguments

<value>::=<NRf> sets the offset of the frequency multi display lines. Range: Center frequency ± Span/2 (Hz)

The default value is the center frequency; the frequency multi display lines are placed from the center frequency at regular intervals.

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:FREQuency:OFFSet 2GHz
```

sets the offset to 2 GHz.

:DISPlay:SPECtrum:MLINe:FREQuency[:STATe](?)

Determines whether to show the frequency multi display lines in thr spectrum view.

Syntax

```
:DISPlay:SPECtrum:MLINe:FREQuency[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:SPECtrum:MLINe:FREQuency[:STATe]?
```

Arguments

OFF or 0 hides the frequency multi display lines.

ON or 1 shows the frequency multi display lines.

Measurement Modes

SARTIME

Examples

```
:DISPlay:SPECtrum:MLINe:FREQuency:STATe ON
```

shows the frequency multi display lines.

:DISPlay:SPECtrum:X[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or frequency, value (left end) in the spectrum view.

Syntax

```
:DISPlay:SPECtrum::X[:SCALe]:OFFSet <freq>
:DISPlay:SPECtrum::X[:SCALe]:OFFSet?
```

Arguments

<freq>::=<NRf> specifies the minimum horizontal value in the spectrum view. The valid range depends on the measurement frequency band setting in the [:SENSe]:FREQuency:BAND command. Refer to Table 2–39 on page 2–271.

Measurement Modes

All

Examples

```
:DISPlay:SPECtrum:X:SCALe:OFFSet 100MHz
```

sets the minimum horizontal value to 100 MHz.

:DISPlay:SPECtrum:X[:SCALe]:PDIVision(?)

Sets or queries the horizontal, or frequency, scale (per division) in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:X[:SCALe]:PDIVision <freq>
:DISPlay:SPECtrum:X[:SCALe]:PDIVision?
```

Arguments

<freq>::=<NRf> specifies the horizontal scale. Refer to Table 2–39 on page 2–271 for the setting range, where the horizontal scale (/div) = span/10.

Measurement Modes

All

Examples

```
:DISPlay:SPECtrum:X:SCALe:PDIVision 100.0E+3
```

sets the horizontal scale to 100 kHz/div.

:DISPlay:SPECtrum:Y[:SCALe]:FIT (No Query Form)

Runs the auto-scale on the spectrum view. The auto-scale automatically sets the start value and scale of the vertical axis to display the whole waveform.

Syntax `:DISPlay:SPECtrum:Y[:SCALe]:FIT`

Arguments None

Measurement Modes All

Examples `:DISPlay:SPECtrum:Y:SCALe:FIT`
runs the auto-scale on the spectrum view.

:DISPlay:SPECtrum:Y[:SCALe]:FULL (No Query Form)

Sets the vertical axis to the default full-scale in the spectrum view.

Syntax `:DISPlay:SPECtrum:Y[:SCALe]:FULL`

Arguments None

Measurement Modes All

Examples `:DISPlay:SPECtrum:Y:SCALe:FULL`
sets the vertical axis to the default full-scale in the spectrum view.

:DISPlay:SPECtrum:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum vertical, or amplitude, value (bottom end) in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:Y[:SCALe]:OFFSet <ampl>
:DISPlay:SPECtrum:Y[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> sets the minimum vertical value. Range: –200 to 0 dBm.

Measurement Modes

All

Examples

```
:DISPlay:SPECtrum:Y:SCALe:OFFSet -100
```

sets the minimum vertical value to –100 dBm.

:DISPlay:SPECtrum:Y[:SCALe]:PDIVision(?)

Sets or queries the vertical, or amplitude, scale (per division) in the spectrum view.

Syntax

```
:DISPlay:SPECtrum:Y[:SCALe]:PDIVision <ampl>
:DISPlay:SPECtrum:Y[:SCALe]:PDIVision?
```

Arguments

<ampl>::=<NRf> specifies the horizontal scale in the spectrum view. Range: 0 to 10 dB/div.

Measurement Modes

All

Examples

```
:DISPlay:SPECtrum:Y:SCALe:PDIVision 10
```

sets the vertical scale to 10 dB/div.

:DISPlay:TFRequency Subgroup

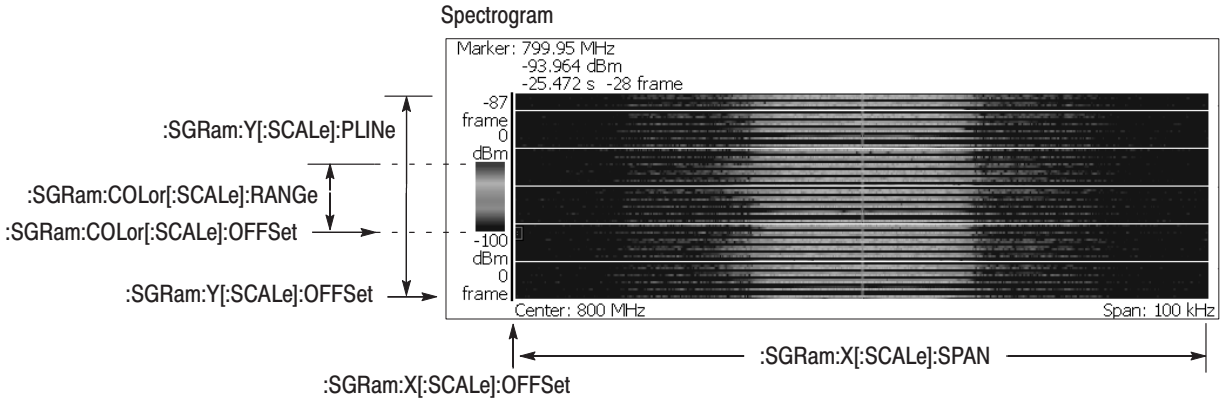
The :DISPlay:TFRequency commands control a three-dimensional view (spectrogram).

***NOTE**. To use a command of this group, you must have selected SARTIME (Real Time S/A) in the :INSTrument[:SELect] command.*

In the SASGRAM (S/A with Spectrogram) mode, you cannot set the scale of the spectrogram.

Command Tree

Header	Parameter
:DISPlay	
:TFRequency	
:SGRam	
:COLor	
[:SCALe]	
:OFFSet	<amplitude>
:RANge	<relative_amplitude>
:MLINe	
:ANNotation	
[:STATe]	<boolean>
:FREQuency	
:INTerval	<numeric_value>
:OFFSet	<numeric_value>
[:STATe]	<boolean>
:TIME	
:INTerval	<numeric_value>
:OFFSet	<numeric_value>
[:STATe]	<boolean>
:X	
[:SCALe]	
:OFFSet	<frequency>
:SPAN	<frequency>
:Y	
[:SCALe]	
:OFFSet	<frame_count>
:PLINe	<frame_count>



NOTE: Command header :DISPlay:TFRequency is omitted here.

Figure 2–12: :DISPlay:TFRequency command setting

:DISPlay:TFRequency:SGRam:COLor[:SCALe]:OFFSet(?)

Sets or queries the minimum value (bottom end) of the color, or amplitude, axis in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:OFFSet <ampl>
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> specifies the minimum color-axis value.
Range: –200 to 0 dBm.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:COLor:SCALe:OFFSet -100
```
sets the minimum color-axis value to –100 dBm.

:DISPlay:TFRequency:SGRam:COLor[:SCALe]:RANGe(?)

Sets or queries full-scale of the color, or amplitude, axis in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:RANGe <rel_ampl>
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:RANGe?
```

Arguments

<rel_ampl>::={ 10 | 20 | 50 | 100 } [dB] specifies full-scale of the color axis.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:COLor:SCALe:RANGe 100
```
sets full-scale of the color axis to 100 dB.

:DISPlay:TFRequency:SGRam:MLINe:ANNotation[:STATe](?)

Determines whether to show the multi display lines readout in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:ANNotation[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:TFRequency:SGRam:MLINe:ANNotation[:STATe]?
```

Arguments

OFF or 0 hides the multi display lines readout.

ON or 1 shows the multi display lines readout.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:ANNotation:STATe ON
```
shows the readout.

:DISPlay:TFRequency:SGRam:MLINe:FREQuency:INTerval(?)

Sets or queries the interval of the frequency multi display lines in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:INTerval <value>
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:INTerval?
```

Arguments

<value>::=<NRf> sets the interval of the frequency multi display lines. Range: 0 to full span (Hz).

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:INTerval 1MHz
```
sets the interval to 1 MHz.

:DISPlay:TFRequency:SGRam:MLINe:FREQuency:OFFSet(?)

Sets or queries the offset of the frequency multi display lines in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:OFFSet <value>
```

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:OFFSet?
```

Arguments

<value>::=<NRf> sets the offset of the frequency multi display lines.
Range: Center frequency ± Span/2 (Hz)

The default value is the center frequency; the frequency multi display lines are placed from the center frequency at regular intervals.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:OFFSet 2GHz
```
sets the offset to 2 GHz.

:DISPlay:TFRequency:SGRam:MLINe:FREQuency[:STATe](?)

Determines whether to show the frequency multi display lines in thr spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency[:STATe] { OFF | ON
| 0 | 1 }
```

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency[:STATe]?
```

Arguments

OFF or 0 hides the frequency multi display lines.

ON or 1 shows the frequency multi display lines.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:STATe ON
```
shows the frequency multi display lines.

:DISPlay:TFRequency:SGRam:MLINe:TIME:INTerval(?)

Sets or queries the interval of the time multi display lines in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:TIME:INTerval <value>
:DISPlay:TFRequency:SGRam:MLINe:TIME:INTerval?
```

Arguments

<value>::=<NRf> sets the interval of the time multi display lines.
Range: 0 second minimum.
The maximum value depends on acquired data quantity.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:TIME:INTerval 1m
```

sets the interval to 1 ms.

:DISPlay:TFRequency:SGRam:MLINe:TIME:OFFSet(?)

Sets or queries the offset of the time multi display lines in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:TIME:OFFSet <value>
:DISPlay:TFRequency:SGRam:MLINe:TIME:OFFSet?
```

Arguments

<value>::=<NRf> sets the offset of the time multi display lines.
Range: 0 second maximum (Zero represents the latest frame.)
The minimum value depends on acquired data quantity.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:TIME:OFFSet -500u
```

sets the offset to –500 µs.

:DISPlay:TFRequency:SGRam:MLINe:TIME[:STATe](?)

Determines whether to show the time multi display lines in thr spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:MLINe:TIME[:STATe] { OFF | ON | 0 | 1 }
:DISPlay:TFRequency:SGRam:MLINe:TIME[:STATe]?
```

Arguments

OFF or 0 hides the time multi display lines.

ON or 1 shows the time multi display lines.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:MLINe:TIME:STATe ON
```

shows the time multi display lines.

:DISPlay:TFRequency:SGRam:X[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or frequency, value (left end) in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:X[:SCALe]:OFFSet <freq>
:DISPlay:TFRequency:SGRam:X[:SCALe]:OFFSet?
```

Arguments

<freq>::=<NRf> specifies the minimum horizontal value in the spectrogram. The valid range depends on the measurement frequency band setting in the [:SENSe]:FREQuency:BAND command. Refer to Table 2–39 on page 2–271.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:X:SCALe:OFFSet 100MHz
```

sets the minimum horizontal value to 100 MHz.

Related Commands

[:SENSe]:FREQuency:BAND

:DISPlay:TFRequency:SGRam:X[:SCALe]:SPAN(?)

Sets or queries the horizontal, or frequency, span in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:X[:SCALe]:SPAN <freq>
:DISPlay:TFRequency:SGRam:X[:SCALe]:SPAN?
```

Arguments

<freq>::=<NRf> specifies the horizontal span in the spectrogram.
Range: 100 Hz to 10 MHz (RF)
100 Hz to 20 MHz (baseband with option 05)

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:X:SCALe:SPAN 10MHz
```

sets the span to 10 MHz.

:DISPlay:TFRequency:SGRam:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum horizontal, or frame number, value (bottom end) in the spectrogram.

Syntax

```
:DISPlay:TFRequency:SGRam:Y[:SCALe]:OFFSet <value>
:DISPlay:TFRequency:SGRam:Y[:SCALe]:OFFSet?
```

Arguments

<value>::=<NR1> specifies the minimum vertical value in the spectrogram. Range: Frame # –63999 to 0.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:Y:SCALe:OFFSet -100
```

sets the minimum vertical value to frame # –100.

:DISPlay:TFRequency:SGRam:Y[:SCALe]:PLINe(?)

Sets or queries the vertical scale (the number of frames per line) when the overview displays a spectrogram.

Frames are thinned out from all the acquired framed data at intervals of the number of frames specified in this command, before the spectrogram is displayed. For example, if you set the argument to 5, the data will be displayed every 5 frames.

Syntax

```
:DISPlay:TFRequency:SGRam:Y[:SCALe]:PLINe <value>
:DISPlay:TFRequency:SGRam:Y[:SCALe]:PLINe?
```

Arguments

<value>::=<NR1> specifies the vertical scale in the spectrogram. Range: 1 to 1024 frames per line.

Measurement Modes

SARTIME

Examples

```
:DISPlay:TFRequency:SGRam:Y:SCALe:PLINe 5
```

displays the data in the spectrogram every 5 frames.

:DISPlay[:VIEW] Subgroup

The :DISPlay[:VIEW] commands control the display brightness and format.

Command Tree

Header	Parameter
:DISPlay	
[:VIEW]	
:BRIGhtness	<numeric_value>
:FORMat	V1S | V3S | V4S | VSPL | HSPL | MULTitude

:DISPlay[:VIEW]:BRIGhtness(?)

Sets or queries the display brightness.

Syntax

```
:DISPlay[:VIEW]:BRIGhtness <value>
:DISPlay[:VIEW]:BRIGhtness?
```

Arguments

<value>::=<NRf> specifies the brightness. Range: 0 to 1. One represents the maximum brightness.

Measurement Modes

All

Examples

```
:DISPlay:VIEW:BRIGhtness 1
```

sets the display brightness to 1 (maximum).

:DISPlay[:VIEW]:FORMat(?)

Selects or queries the view display format.

Syntax

```
:DISPlay[:VIEW]:FORMat { V1S | V3S | V4S | VSPL | HSPL
| MULTitude }
:DISPlay[:VIEW]:FORMat?
```

Arguments

V1S specifies that only View 1 is displayed.

V3S specifies that only View 3 is displayed.

V4S specifies that only View 4 is displayed.

VSPL specifies that Views 1 and 4 are tiled horizontally.

HSPL specifies that Views 1 and 4 are tiled vertically.

MULTitude specifies that multiple views are displayed simultaneously.

***NOTE**. You must have selected SASGRAM or SARTIME with the INSTrument[:SELect] command to use VSPL or HSPL.*

You must have selected a measurement mode which has three views to use MULTitude.

Measurement Modes All

Examples `:DISPlay:VIEW:FORMat V1S`
specifies that only View 1 is displayed.

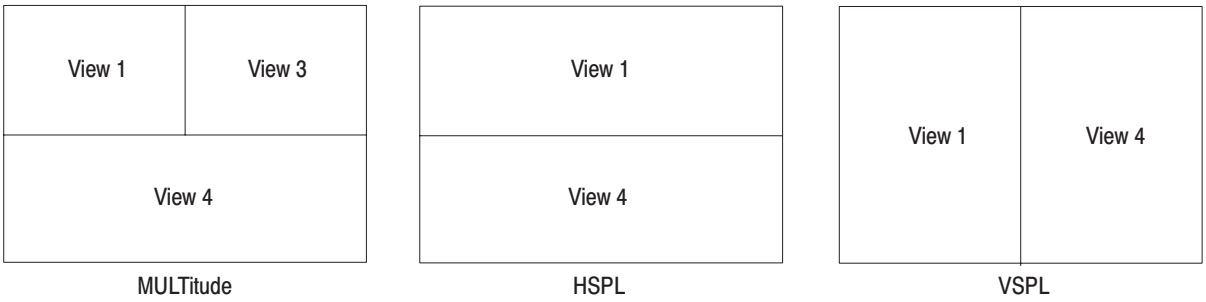


Figure 2–13: View display formats

Related Commands `:INSTrument[:SELect]`

:DISPlay:WAVeform Subgroup

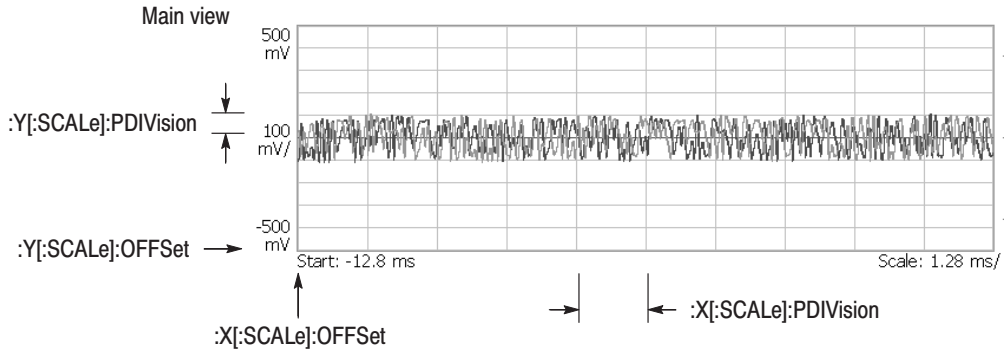
The :DISPlay:WAVeform commands control the time domain display in the main view in the Demod (modulation analysis) and Time (time analysis) modes. There are six types of time domain display associated with six different measurement items:

Frequency vs. Time
I/Q level vs. Time
Frequency shift vs. Time
AM demodulation display (percentage modulation vs. time)
FM demodulation display (frequency shift vs. time)
PM demodulation display (phase shift vs. time)

***NOTE**. To use a command of this group, you must have selected DEMADEM (analog modulation analysis) or TIMTRAN (time characteristic analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
:DISPlay	
:WAVeform	
:X	
[:SCALe]	
:OFFSet	<time>
:PDIVsion	<time>
:Y	
[:SCALe]	
:FIT	
:FULL	
:OFFSet	<amplitude>
:PDIVsion	<amplitude>



NOTE: Command header :DISPlay:WAVeform is omitted here.

Figure 2–14: :DISPlay:WAVeform command setting

:DISPlay:WAVeform:X[:SCALe]:OFFSet(?)

Sets or queries the minimum value of the horizontal axis (left end) in the time domain display.

Syntax

```
:DISPlay:WAVeform:X[:SCALe]:OFFSet <time>
:DISPlay:WAVeform:X[:SCALe]:OFFSet?
```

Arguments

<time>::=<NRf> sets the minimum horizontal value. Range: –32000 to 0 s.

Measurement Modes

DEMADEM, TIMTRAN

Examples

```
:DISPlay:WAVeform:X:SCALe:OFFSet -100us
```

sets the minimum horizontal value to –100 μs.

:DISPlay:WAVeform:X[:SCALe]:PDIVision(?)

Sets or queries the horizontal, or time, scale (per division) in the time domain display.

Syntax

```
:DISPlay:WAVeform:X[:SCALe]:PDIVision <time>
:DISPlay:WAVeform:X[:SCALe]:PDIVision?
```

Arguments

<time>::=<NRf> specifies the horizontal scale. Range: 0 to 3200 s/div

Measurement Modes

DEMADEM, TIMTRAN

Examples

```
:DISPlay:WAVeform:X:SCALe:PDIVision 10us
```

sets the horizontal scale to 10 μs/div.

:DISPlay:WAVeform:Y[:SCALe]:FIT (No Query Form)

Runs the auto-scale on the time domain display. The auto-scale automatically sets the start value and scale of the vertical axis to display the whole waveform.

Syntax `:DISPlay:WAVeform:Y[:SCALe]:FIT`

Arguments None

Measurement Modes `DEMADEM`, `TIMTRAN`

Examples `:DISPlay:WAVeform:Y:SCALe:FIT`
runs the auto-scale.

:DISPlay:WAVeform:Y[:SCALe]:FULL (No Query Form)

Sets the vertical axis in the time domain display to the default full-scale.

Syntax `:DISPlay:WAVeform:Y[:SCALe]:FULL`

Arguments None

Measurement Modes `DEMADEM`, `TIMTRAN`

Examples `:DISPlay:WAVeform:Y:SCALe:FULL`
sets the vertical axis in the time domain display to the default full-scale.

:DISPlay:WAVeform:Y[:SCALe]:OFFSet(?)

Sets or queries the minimum value of the vertical axis (bottom end) in the time domain display.

Syntax

```
:DISPlay:WAVeform:Y[:SCALe]:OFFSet <ampl>
:DISPlay:WAVeform:Y[:SCALe]:OFFSet?
```

Arguments

<ampl>::=<NRf> specifies the minimum value of the vertical axis. The valid range depends on the display format. Refer to Table D–1 in *Appendix D*.

Measurement Modes

DEMADEM, TIMTRAN

Examples

```
:DISPlay:WAVeform:Y:SCALe:OFFSet -100
```
sets the minimum vertical value to –100 dBm.

:DISPlay:WAVeform:Y[:SCALe]:PDIVision(?)

Sets the vertical axis scale (per division) in the time domain display.

Syntax

```
:DISPlay:WAVeform:Y[:SCALe]:PDIVision <ampl>
:DISPlay:WAVeform:Y[:SCALe]:PDIVision?
```

Arguments

<ampl>::=<NRf> specifies the vertical scale. The valid range depends on the display format. Refer to Table D–1 in *Appendix D*.

Measurement Modes

DEMADEM, TIMTRAN

Examples

```
:DISPlay:WAVeform:Y:SCALe:PDIVision 10
```
sets the vertical scale to 10 dB/div

:FETCh Commands

The :FETCh commands retrieve the measurements from the data taken by the latest INITiate command.

If you want to perform a FETCh operation on fresh data, use the :READ commands on page 2–207. The :READ commands acquire a new input signal and fetch the measurement results from that data.

***NOTE**. To use a :FETCh command, you must have set a measurement mode for the FETCh operation using the :INSTrument[:SELect] command (refer to page 2–191).*

Command Tree

```
Header                  Parameter
:FETCh
   :ADEMod
      :AM?
         :RESult?
      :FM?
         :RESult?
      :PM?
      :PSPectrum?
   :CCDF?
   :DISTribution:CCDF?
   :OVIew?
   :PULSe?              ALL | WIDTh | PPOWer | OORatio | RIPPle
                        | PERiod | DCYCle | PHASe | CHPower
                        | OBWidth | EBWidth | FREQuency
      :SPECtrum?
      :TAMPlitude?
      :TFRequency?
   :SPECtrum?
      :ACPower?
      :CFRequency?
      :CHPower?
      :CNRatio?
      :EBWidth?
      :OBWidth?
      :SPURious?
   :TRANsient
      :FVTime?
      :IQVTime?
      :PVTime?
```

:FETCh:ADEMod:AM? (Query Only)

Returns the results of the AM signal analysis in time series.

Syntax `:FETCh:ADEMod:AM?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the percentage modulation data in percent (%) for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes DEMADEM

Examples `:FETCh:ADEMod:AM?`
might return `#41024xxxx...` (1024-byte data) for the results of the AM signal analysis.

Related Commands `:INSTrument[:SELect]`

:FETCh:ADEMod:AM:RESult? (Query Only)

Returns the measurement results of the AM signal analysis.

Syntax `:FETCh:ADEMod:AM:RESult?`

Arguments None

Returns `<+AM>,<–AM>,<Total_AM>`

Where
`<+AM>::=<NRf>` is the positive peak AM value in percent (%).
`<–AM>::=<NRf>` is the negative peak AM value in percent (%).
`<Total_AM>::=<NRf>` is the total AM value: (peak-peak AM value) / 2 in percent (%).

Measurement Modes `DEMADEM`

Examples `:FETCh:ADEMod:AM:RESult?`
might return `37.34,–48.75,43.04`.

Related Commands `:INSTrument[:SELect]`

:FETCh:ADEMod:FM? (Query Only)

Returns the results of the FM signal analysis in time series.

Syntax `:FETCh:ADEMod:FM?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the frequency shift data in Hz for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes DEMADEM

Examples `:FETCh:ADEMod:FM?`
might return `#41024xxxx...` (1024-byte data) for the results of the FM signal analysis.

Related Commands `:INSTrument[:SELect]`

:FETCh:ADEMod:FM:RESult? (Query Only)

Returns the measurement results of the FM signal analysis.

Syntax `:FETCh:ADEMod:FM:RESult?`

Arguments None

Returns `<+Pk_Freq_Dev>,<-Pk_Freq_Dev>,<P2P_Freq_Dev>,<P2P_Freq_Dev/2>,<RMS_Freq_Dev>`

Where
`<+Pk_Freq_Dev>::=<NRf>` is the positive peak frequency deviation in Hz.
`<-Pk_Freq_Dev>::=<NRf>` is the negative peak frequency deviation in Hz.
`<P2P_Freq_Dev>::=<NRf>` is the peak-to-peak frequency deviation in Hz.
`<P2P_Freq_Dev/2>::=<NRf>` is (peak-to-peak frequency deviation) / 2 in Hz.
`<RMS_Freq_Dev>::=<NRf>` is the RMS frequency deviation in Hz.

Examples `:FETCh:ADEMod:FM:RESult?`
might return `1.13e+4,-1.55e+4,2.48e+4,1.24e+4,1.03e+4`.

Related Commands `:INSTrument[:SELect]`

:FETCh:ADEMod:PM? (Query Only)

Returns the results of the PM signal analysis in time series.

Syntax `:FETCh:ADEMod:PM?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the phase shift data in degree for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes DEMADEM

Examples `:FETCh:ADEMod:PM?`
might return `#41024xxxx...` (1024-byte data) for the results of the PM signal analysis.

Related Commands `:INSTrument[:SELect]`

:FETCh:ADEMod:PSPectrum? (Query Only)

Returns spectrum data of the pulse spectrum measurement in the analog modulation analysis.

Syntax

```
:FETCh:ADEMod:PSPectrum?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Data(n)> is the spectrum amplitude in dBm.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 240001

Measurement Modes

DEMADEM

Examples

```
:FETCh:ADEMod:PSPectrum?
```

might return `#43200xxxx...` (3200-byte data) for the spectrum data.

Related Commands

```
:INSTrument[:SELect]
```

:FETCh:CCDF? (Query Only)

Returns the CCDF measurement results.

Syntax `:FETCh:CCDF?`

Arguments None

Returns `<meanpower>,<peakpower>,<cfactor>`

Where
`<meanpower>::=<NRf>` is the average power measured value in dBm.
`<peakpower>::=<NRf>` is the peak power measured value in dBm.
`<cfactor>::=<NRf>` is the crest factor in dB.

Measurement Modes `TIMCCDF`

Examples `:FETCh:CCDF?`
might return `-11.16,-8.18,2.96` for the CCDF measurement results.

Related Commands `:INSTrument[:SELect]`

:FETCh:DISTribution:CCDF? (Query Only)

Returns the CCDF trace data in the CCDF measurement.

Syntax

```
:FETCh:DISTribution:CCDF?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the phase shift data in degrees for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 10001

Invalid data is returned as –1000.

Measurement Modes

TIMCCDF

Examples

```
:FETCh:DISTribution:CCDF?
```

might return `#41024xxxx...` (1024-byte data) for the CCDF trace data in the CCDF measurement.

Related Commands

:FETCh:CCDF?, :INSTrument[:SELect]

:FETCh:OVIew? (Query Only)

Returns the minimum and maximum values for each 1024-point segment of waveform data displayed on the overview in the Demod (modulation analysis) and the Time (time analysis) modes.

***NOTE**. The :CONFigure:OVIew command must be run to turn measurement off before the :FETCh:OVIew command is executed.*

Syntax

```
:FETCh:OVIew?
```

Returns

```
#<Num_digit><Num_byte><MinData(1)><MaxData(1)>...
<MinData(n)><MaxData(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<MinData(n)> is the minimum data in dBm for each 1024 data point segment.
4-byte little endian floating-point format specified in IEEE 488.2
<MaxData(n)> is the maximum data in dBm for each 1024 data point segment.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 16000 (standard) / 64000 (Option 02)

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:FETCh:OVIew?
```

might return #510240xxx... (10240-byte data) representing the minimum and the maximum values of waveform displayed on the overview.

Related Commands

:CONFigure:OVIew, :INSTrument[:SELect]

:FETCh:PULSe? (Query Only)

Returns the result of the pulse characteristics analysis.

Syntax

```
:FETCh:PULSe? { ALL | WIDTh | PPOWer | OORatio | RIPPle | PERiod
| DCYCle | PHASe | CHPower | OBWidth | EBWidth | FREQuency }
```

Arguments

Information queried is listed below for each of the arguments:

Table 2–32: Queried information

Argument	Information queried
ALL	All
WIDTh	Pulse width
PPOWer	Maximum (peak) power in the pulse on-time
OORatio	Difference between the on-time power and off-time power
RIPPle	Difference between the maximum and the minimum power in the on-time
PERiod	Time between the pulse rising edge and the next rising edge
DCYCle	Ratio of the pulse width to teh pulse repetition interval (PRI)
PHASe	Phase at a certain point of each pulse
CHPower	Channel power of the pulse on-time spectrum
OBWidth	OBW (Occupied Bandwidth) of the pulse on-time spectrum
EBWidth	EBW (Emission Bandwidth) of the pulse on-time spectrum
FREQuency	Frequency deviation in the pulse on-time

Returns Returns are listed below for each of the arguments.

ALL. <width>,<ppower>,<ooratio>,<ripple>,<period>,<dcycle>,<phase>,<chp>,<obw>,<ebw>,<freq>

Where
<width>::=<NRf> is the pulse width in s.
<ppower>::=<NRf> is the peak power in watts.
<ooratio>::=<NRf> is the on/off ratio in dB.
<ripple>::=<NRf> is the pulse ripple in watts.
<period>::=<NRf> is the pulse repetition interval in s.
<dcycle>::=<NRf> is the duty cycle in percent (%).
<phase>::=<NRf> is the pulse-pulse phase in degrees.
<chp>::=<NRf> is the channel power in watts.
<obw>::=<NRf> is the OBW in Hz.
<ebw>::=<NRf> is the EBW in Hz.
<freq>::=<NRf> is the frequency deviation in Hz.

WIDTh. #<Num_digit><Num_byte><Width(1)><Width(2)>...<Width(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Width(n)> is the pulse width value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

PPOWer. #<Num_digit><Num_byte><Ppower(1)><Ppower(2)>...<Ppower(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Power(n)> is the peak power value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

OORatio. #<Num_digit><Num_byte><Ooratio(1)><Ooratio(2)>...<Ooratio(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Ooratio(n)> is the on/off ratio value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

RIPPle. #<Num_digit><Num_byte><Ripple(1)><Ripple(2)>...<Ripple(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Ripple(n)> is the ripple value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

PERiod. #<Num_digit><Num_byte><Period(1)><Period(2)>...<Period(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Period(n)> is the pulse repetition interval value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

DCYCle. #<Num_digit><Num_byte><Dcycle(1)><Dcycle(2)>...<Dcycle(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Dcycle(n)> is the duty value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

PHASe. #<Num_digit><Num_byte><Phase(1)><Phase(2)>...<Phase(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Phase(n)> is the pulse–pulse phase value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

CHPower. #<Num_digit><Num_byte><Chp(1)><Chp(2)>...<Chp(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Chp(n)> is the Channel Power value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

OBWidth. #<Num_digit><Num_byte><Obw(1)><Obw(2)>...<Obw(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Obw(n)> is the OBW value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

EBWidth. #<Num_digit><Num_byte><Ebw(1)><Ebw(2)>...<Ebw(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Ebw(n)> is the EBW value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

FREQuency. #<Num_digit><Num_byte><Freq(1)><Freq(2)>...<Freq(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Freq(n)> is the frequency deviation value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

Measurement Modes

TIMPULSE

Examples

:FETCh:PULSe? WIDTh
might return #3500xxxx... (500-byte data) for the pulse width measurement result.

Related Commands

:INSTrument[:SELect]

:FETCh:PULSe:SPECtrum? (Query Only)

Returns the spectrum data of the frequency domain measurement in the pulse characteristics analysis.

This query command is valid when :DISPlay:PULSe:SVIew:FORMat is set to CHPowr, OBWidth, or EBWidth.

Syntax

```
:FETCh:PULSe:SPECtrum?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Data(n)> is the spectrum in dBm.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 16384

Measurement Modes

TIMPULSE

Examples

```
:FETCh:PULSe:SPECtrum?
```

might return #43200xxxx... (3200-byte data) for the spectrum data.

Related Commands

:DISPlay:PULSe:SVIew:FORMat, :INSTrument[:SELect]

:FETCh:PULSe:TAMPlitude? (Query Only)

Returns the time domain amplitude data of the time domain measurement in the pulse characteristics analysis.

This query command is valid when :DISPlay:PULSe:SVIew:FORMat is set to WIDTh, PPOWer, OORatio, RIPPle, PERiod, DCYCle, or PHASe.

Syntax `:FETCh:PULSe:TAMPlitude?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
`<Num_digit>` is the number of digits in <Num_byte>.
`<Num_byte>` is the number of bytes of data that follow.
`<Data(n)>` is the absolute power for each data in watts.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 262,144

Measurement Modes `TIMPULSE`

Examples `:FETCh:PULSe:TAMPlitude?`
might return `#43200xxxx...` (3200-byte data) for the time domain amplitude.

Related Commands `:DISPlay:PULSe:SVIew:FORMat`, `:INSTrument[:SELect]`

:FETCh:PULSe:TFRequency? (Query Only)

Returns the frequency deviation measurement results in the pulse characteristics analysis.

This query command is valid when :DISPlay:PULSe:SVIew:FORMat is set to FREQuency.

Syntax

```
:FETCh:PULSe:TFRequency?
```

Arguments

None

Returns

```
#<Num_digit><Num_byre><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Data(n)> is the frequency deviation value in Hz on the time axis.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 262,144

Measurement Modes

TIMPULSE

Examples

```
:FETCh:PULSe:TFRequency?
```

might return #43200xxxx... (3200-byte data) for the time domain frequency.

Related Commands

:DISPlay:PULSe:SVIew:FORMat, :INSTrument[:SELect]

:FETCh:SPECtrum? (Query Only)

Returns spectrum waveform data in the S/A (spectrum analysis) mode.

Syntax `:FETCh:SPECtrum?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the amplitude spectrum in dBm.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 240001

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:FETCh:SPECtrum?`
might return `#43200xxxx...` (3200-byte data) for the spectrum waveform data.

Related Commands `:INSTrument[:SELect]`

:FETCh:SPECtrum:ACPower? (Query Only)

Returns the results of adjacent channel leakage power ratio (ACPR) measurement in the S/A (spectrum analysis) mode.

Syntax

```
:FETCh:SPECtrum:ACPower?
```

Arguments

None

Returns

```
<chpower>,<acpm1>,<acpp1>,<acpm2>,<acpp2>,<acpm3>,<acpp3>
```

Where
<chpower>::=<NRf> is the channel power measured value in dBm.
<acpm1>::=<NRf> is the first lower adjacent channel ACPR in dB.
<acpp1>::=<NRf> is the first upper adjacent channel ACPR in dB.
<acpm2>::=<NRf> is the second lower adjacent channel ACPR in dB.
<acpp2>::=<NRf> is the second upper adjacent channel ACPR in dB.
<acpm3>::=<NRf> is the third lower adjacent channel ACPR in dB.
<acpp3>::=<NRf> is the third upper adjacent channel ACPR in dB.

***NOTE**. All the values may not be returned when the adjacent channel(s) goes out of the span due to the settings of the channel bandwidth and spacing (refer to the [:SENSe]:ACPower subgroup). For example, if the third adjacent channel goes out of the span, the response is <chpower>,<acpm1>,<acpp1>,<acpm2>, <acpp2>; <acpm3> and <acpp3> are not returned.*

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:FETCh:SPECtrum:ACPower?
```

might return `-11.38,-59.41,-59.51,-59.18,-59.31,-59.17,-59.74` for the ACPR measurement results.

Related Commands

:INSTrument[:SELect], [:SENSe]:ACPower subgroup

:FETCh:SPECtrum:CFRequency? (Query Only)

Returns the results of the carrier frequency measurement in the S/A (spectrum analysis) mode.

Syntax `:FETCh:SPECtrum:CFRequency?`

Arguments None

Returns `<cfreq>::=<NRf>` is the measured value of carrier frequency in Hz.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:FETCh:SPECtrum:CFRequency?`
might return `846187328.5` for the carrier frequency.

Related Commands `:INSTrument[:SELect]`

:FETCh:SPECtrum:CHPower? (Query Only)

Returns the results of the channel power measurement in the S/A (spectrum analysis) mode.

Syntax `:FETCh:SPECtrum:CHPower?`

Arguments None

Returns `<chpower>::=<NRf>` is the channel power measured value in dBm.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:FETCh:SPECtrum:CHPower?`
might return `-1.081` for the measurement results of channel power.

Related Commands `:INSTrument[:SELect]`

:FETCh:SPECtrum:CNRatio? (Query Only)

Returns the results of the carrier-to-noise ratio (C/N) measurement in the S/A (spectrum analysis) mode.

Syntax

```
:FETCh:SPECtrum:CNRatio?
```

Arguments

None

Returns

```
<ctn>,<ctno>
```

Where
<ctn>::=<NRf> is the measured value of C/N in dB.
<ctno>::=<NRf> is the measured value of C/No in dB/Hz.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:FETCh:SPECtrum:CNRatio?
```

might return 75.594,125.594 for the C/N measurement results.

Related Commands

:INSTrument[:SELect]

:FETCh:SPECtrum:EBWidth? (Query Only)

Returns the results of the emission bandwidth (EBW) measurement in the S/A (spectrum analysis) mode.

Syntax `:FETCh:SPECtrum:EBWidth?`

Arguments None

Returns `<ebw>::=<NRf>` is the measured value of EBW in Hz.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:FETCh:SPECtrum:EBWidth?`
might return `30956.26` for the EBW measurement results.

Related Commands `:INSTrument[:SELect]`

:FETCh:SPECtrum:OBWidth? (Query Only)

Returns the results of the occupied bandwidth (OBW) measurement in the S/A (spectrum analysis) mode.

Syntax `:FETCh:SPECtrum:OBWidth?`

Arguments None

Returns `<obw>::=<NRf>` is the measured value of OBW in Hz.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:FETCh:SPECtrum:OBWidth?`
might return `26510.163` for the OBW measurement results.

Related Commands `:INSTrument[:SELect]`

:FETCh:SPECtrum:SPURious? (Query Only)

Returns the results of the spurious signal measurement in the S/A (spectrum analysis) mode.

Syntax

```
:FETCh:SPECtrum:SPURious?
```

Arguments

None

Returns

```
<snum>{,<dfreq>,<rdb>}
```

Where
<snum>::=<NR1> is the number of detected spurious emissions, max. 20
<dfreq>::=<NRf> is the detuned frequency of spurious relative to carrier in Hz.
<rdb>::=<NRf> is the spurious signal level relative to carrier in dB.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:FETCh:SPECtrum:SPURious?
```

might return `3,1.2E6,-79,2.4E6,-79.59,1E6,-80.38` for the spurious signal measurement.

Related Commands

:INSTrument[:SELect]

:FETCh:TRANsient:FVTime? (Query Only)

Returns the results of the frequency vs. time measurement in the Time (time analysis) mode.

Syntax `:FETCh:TRANsient:FVTime?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the frequency data in Hz for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes `TIMTRAN`

Examples `:FETCh:TRANsient:FVTime?`
might return `#41024xxxx...` (1024-byte data) for the results of the frequency vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:FETCh:TRANsient:IQVTime? (Query Only)

Returns the results of the IQ level vs. time measurement in the Time (time analysis) mode.

Syntax

```
:FETCh:TRANsient:IQVTime?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Idata(1)><Qdata(1)>
<Idata(2)><Qdata2>...<Idata(n)><Qdata(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Idata(n)><Qdata(n)> is the I and Q signal level data in volt for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points x 500 frames)

Measurement Modes

```
TIMTRAN
```

Examples

```
:FETCh:TRANsient:IQVTime?
```

might return `#41024xxxx...` (1024-byte data) for the results of the IQ level vs. time measurement.

Related Commands

```
:INSTrument[:SELect]
```

:FETCh:TRANsient:PVTime? (Query Only)

Returns the results of the power vs. time measurement in the Time (time analysis) mode.

Syntax `:FETCh:TRANsient:PVTime?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the time domain power data in dBm.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes TIMTRAN

Examples `:FETCh:TRANsient:PVTime?`
might return `#41024xxxx...` (1024-byte data) for the results of the power vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:FORMat Commands

The FORMat commands define the data output format.

Command Tree

Header	Parameter
:FORMat	
:BORDer	NORMal | SWAPped
[:DATA]	REAL,32 | REAL,64

:FORMat:BORDer (?)

Sets or queries the byte order for transferring binary data.

Syntax

```
:FORMat:BORDer { NORMal | SWAPped }
:FORMat:BORDer?
```

Arguments

`NORMal` selects the normal byte order.

`SWAPped` swaps the byte order.

Measurement Modes

All

Examples

```
:FORMat:BORDer SWAPped
```
swaps the byte order for data output.

:FORMat[:DATA] (?)

Selects or queries the output data format.

Syntax

```
:FORMat[:DATA] { REAL,32 | REAL,64 }
:FORMat[:DATA]?
```

Arguments

`REAL,32` specifies the 32-bit floating point format.

`REAL,64` specifies the 64-bit floating point format.

Measurement Modes

All

Examples

```
:FORMat:DATA REAL,32
```
specifies the 32-bit floating point format for data output.

:HCOPy Commands

The :HCOPy commands control screen hardcopy.

Command Tree

Header	Parameter
:HCOPy	
:BACKground	BLACk | WHITe
:DESTination	PRINter | MMEMory
[:IMMediate]	

:HCOPy:BACKground (?)

Selects or queries the hardcopy background color.

Syntax

```
:HCOPy:BACKground { BLACk | WHITe }
:HCOPy:BACKground?
```

Arguments

BLACK outputs the screen image in the black background, without reversing it.

WHITe reverses the screen image to output it in the white background.

Measurement Modes

All

Examples

```
:HCOPy:BACKground WHITe
```

reverses the screen image to output it in the white background.

:HCOPy:DESTination (?)

Selects or queries the hardcopy output destination (printer or file).

Syntax

```
:HCOPy:DESTination { PRINter | MMEMory }
:HCOPy:DESTination?
```

Arguments

PRINTer specifies that the hardcopy is output to the preset printer, which is the one that has been set as the printer to be used usually under Windows. For using the printer, refer to the *RSA2203A and RSA2208A User Manual*.

MMEMory specifies that the hardcopy is output to the bitmap file specified with the :MMEMory:NAME command.

Measurement Modes

All

Examples

```
:HCOPy:DESTination PRINter
```

specifies that the hardcopy is output to the preset printer.

Related Commands

:HCOPy[:IMMediate], :MMEMory:NAME

:HCOPy[:IMMediate] (No Query Form)

Outputs the screen hardcopy to the destination selected with the :HCOPy:DESTination command.

Syntax `:HCOPy[:IMMediate]`

Arguments None

Measurement Modes All

Examples `:HCOPy:IMMediate`
outputs the screen hardcopy.

Related Commands `:HCOPy:DESTination`

:INITiate Commands

The :INITiate commands control data acquisition.

Command Tree

Header	Parameter
:INITiate	
:CONTinuous	<boolean>
[:IMMediate]	
:RESTart	

:INITiate:CONTinuous (?)

Determines whether to use the continuous mode to acquire the input signal.

Syntax

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

Arguments

OFF or 0 specifies that the single mode, rather than the continuous mode, is used for data acquisition. To initiate the acquisition, use the :INITiate[:IMMediate], described below.

To stop the acquisition because the trigger is not generated in single mode, send the following command:

```
:INITiate:CONTinuous OFF
```

ON or 1 initiates data acquisition in the continuous mode.

To stop the acquisition in the continuous mode, send the following command:

```
:INITiate:CONTinuous OFF
```

***NOTE**. When the analyzer receives a :FETCh command while operating in the continuous mode, it returns an execution error. If you want to run a :FETCh, use the :INITiate[:IMMediate] command.*

Measurement Modes

All

Examples

```
:INITiate:CONTinuous ON
```
specifies that the continuous mode is used to acquire the input signal.

Related Commands

:FETCh commands, :INITiate[:IMMediate]

:INITiate[:IMMediate] (No Query Form)

Starts input signal acquisition.

Syntax `:INITiate[:IMMediate]`

Arguments None

Measurement Modes All

Examples `:INITiate:IMMediate`
Starts input signal acquisition.

Related Commands `:INITiate:CONTinuous`

:INITiate:RESTart (No Query Form)

Reruns input signal acquisition. In the single mode, this command is equivalent to the :INITiate[:IMMediate] command. In the continuous mode, this command is equivalent to the :ABORt command.

Syntax `:INITiate:RESTart`

Arguments None

Measurement Modes All

Examples `:INITiate:RESTart`
reruns input signal acquisition.

Related Commands `:ABORt`, `:INITiate[:IMMediate]`

:INPut Commands

The :INPut commands control the characteristics of the signal input.

Command Tree

Header	Parameter
:INPut	
:ALEVel	
:ATTenuation	<numeric_value>
:AUTO	<boolean>
:MIXer	<numeric_value>
:MLEVel	<numeric_value>

:INPut:ALEVel (No Query Form)

Adjusts amplitude automatically for the best system performance using the input signal as a guide.

Syntax `:INPut:ALEVel`

Arguments None

Measurement Modes All

Examples `:INPut:ALEVel`
adjusts amplitude automatically.

:INPut:ATTenuation (?)

When you have selected OFF or 0 in the :INPut:ATTenuation:AUTO command, described below, use this command to set the input attenuation. The query version of this command returns the input attenuation setting.

Syntax `:INPut:ATTenuation <rel_ampl>`

`:INPut:ATTenuation?`

Arguments `<rel_ampl>::=<NR1>` specifies the input attenuation.
Range: 0 to 50 dB in 10 dB steps.

Measurement Modes All

Examples `:INPut:ATTenuation 20`
sets the input attenuation to 20 dB.

Related Commands `:INPut:ATTenuation:AUTO`

:INPut:ATTenuation:AUTO (?)

Determines whether to automatically set the input attenuation according to the reference level.

Syntax

```
:INPut:ATTenuation:AUTO { OFF | ON | 0 | 1 }
```

```
:INPut:ATTenuation:AUTO?
```

Arguments

OFF or 0 specifies that the input attenuation is not set automatically. To set it, use the :INPut:ATTenuation command, described above.

ON or 1 specifies that the input attenuation is set automatically.

Measurement Modes

All

Examples

```
:INPut:ATTenuation:AUTO ON
```

specifies that the input attenuation is set automatically.

Related Commands

```
:INPut:ATTenuation
```

:INPut:MIXer (?)

Selects or queries the mixer level.

***NOTE**. To set the mixer level, you must have selected On in the :INPut:ATTenuation:AUTO command.*

Syntax

```
:INPut:MIXer <ampl>
:INPut:MIXer?
```

Arguments

<ampl>::=<NR1> specifies the mixer level. The valid settings depend on the measurement frequency band as shown in Table 2–33.

Table 2–33: Mixer level settings

Measurement frequency band	Setting (dBm)
RF (RSA2203A) / RF1 (RSA2208A)	–5, –10, –15, –20, or –25
RF2, RF3 (RSA2208A)	–5, –15, or –25

Measurement Modes

All

Examples

```
:INPut:MIXer -20
```
sets the mixer level to –20 dBm.

Related Commands

```
:INPut:ATTenuation:AUTO
```

:INPut:MLEVel (?)

Sets or queries the reference level. Using this command to set the reference level is equivalent to pressing the **AMPLITUDE** key and then the **Ref Level** side key on the front panel.

Syntax

```
:INPut:MLEVel <ampl>
:INPut:MLEVel?
```

Arguments

<ampl>::=<NR1> specifies the reference level. The valid settings depend on the measurement frequency band as shown in Table 2–34.

Table 2–34: Reference level range

Measurement frequency band	Setting
RF (RSA2203A) / RF1 (RSA2208A)	–51 to +30 dBm (in 1 dB steps)
RF2, RF3 (RSA2208A)	–50 to +30 dBm (in 1 dB steps)
Baseband (Option 05)	–30 to +20 dBm (in 2 dB steps)

Measurement Modes

All

Examples

```
:INPut:MLEVel -10
```

sets the reference level to –10 dBm.

:INSTrument Commands

The :INSTrument commands set the measurement mode. Before you can start a measurement, you must set the mode appropriate for the measurement using these commands.

Command Tree

Header	Parameter
:INSTrument	
:CATalog?	
[:SELect]	<mode_name>

:INSTrument:CATalog? (Query Only)

Queries all the measurement modes incorporated in the analyzer.

Syntax

```
:INSTrument:CATalog?
```

Arguments

None

Returns

<string> contains the measurement mode names available in the analyzer returned as comma-separated character strings. The following table lists the mode names and their meanings:

Table 2–35: Measurement mode

Mnemonic	Meaning
SANORMAL	Normal spectrum analysis
SASGRAM	Spectrum analysis with spectrogram
SARTIME	Real-time spectrum analysis
SAZRTIME	Real-time spectrum analysis with zoom function
DEMADEM	Analog modulation analysis
TIMCCDF	CCDF analysis
TIMTRAN	Time characteristic analysis
TIMPULSE	Pulse characteristics analysis

In the full options case, all the above mode names are returned as comma-separated character strings.

Measurement Modes

All

Examples

```
:INSTrument:CATalog?
```

might return `"SANORMAL","SASGRAM","SARTIME","DEMADEM","TIMCCDF","TIMTRAN"` for all the measurement modes that the analyzer has.

:INSTrument[:SELect] (?)

Selects or queries the measurement mode.

This command is not affected by *RST.

***NOTE**. If you want to change the measurement mode, stop the data acquisition with the :INITiate:CONTinuous OFF command.*

Syntax

```
:INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME | SAZRTIME
| DEMADEM | TIMCCDF | TIMTRAN | TIMPULSE }

:INSTrument[:SELect]?
```

Arguments

`<string>`

For details of the modes, refer to Table 2–35 on the previous page.

Examples

```
:INSTrument:SELect "DEMADEM"
```

places the analyzer in the analog modulation analysis mode.

Related Commands

`:CONFigure`, `:INITiate:CONTinuous`

:MMEMory Commands

The :MMEMory commands allow you to manipulate files on the hard disk or floppy disk.

For details on file manipulation, refer to the *RSA2203A and RSA2208A User Manual*.

Command Tree

Header	Parameter
:MMEMory	
:COPY	<file_name1>,<file_name2>
:DELete	<file_name>
:LOAD	
:CORRection	<file_name>
:IQT	<file_name>
:STATe	<file_name>
:TRACe	<file_name>
:NAME	<file_name>
:STORe	
:CORRection	<file_name>
:IQT	<file_name>
:PULSe	<file_name>
:STATe	<file_name>
:TRACe	<file_name>

***NOTE**. Use the absolute path to specify the file name. For example, suppose that data file Sample1.iqt is located in the My Documents folder of Windows. You can specify it as "C:\My Documents\Sample1.iqt."*

:MMEMory:COPY (No Query Form)

Copies the contents of a file to another.

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

Arguments `<file_name1>::=<string>` specifies the source file.

`<file_name2>::=<string>` specifies the destination file.

Measurement Modes All

Examples `:MMEMory:COPY "C:\My Documents\File1","C:\My Documents\File2"`
copies the contents of File1, located in the My Documents folder, to File2.

:MMEMory:DELete (No Query Form)

Deletes the specified file.

Syntax `:MMEMory:DELete <file_name>`

Arguments `<file_name>::=<string>` specifies the file to be deleted.

Measurement Modes All

Examples `:MMEMory:DELete "C:\My Documents\File1"`
deletes File1 located in the My Documents folder.

:MMEMory:LOAD:CORRection (No Query Form)

Loads the amplitude correction file.

Syntax `:MMEMory:LOAD:CORRection <file_name>`

Arguments `<file_name>::=<string>` specifies the file that contains the amplitude correction table. The file extension is .cor.

Measurement Modes SANORMAL, SASGRAM

Examples `:MMEMory:LOAD:CORRection "C:\My Documents\File1.cor"`
loads the correction table from File1.cor in the My Documents folder.

:MMEMory:LOAD:IQT (No Query Form)

Loads IQ data in time domain from the specified file.

Syntax `:MMEMory:LOAD:IQT <file_name>`

Arguments `<file_name>::=<string>` specifies the file from which to load IQ data. The file extension is .iqt.

Measurement Modes SARTIME, DEMADEM, TIMCCDF, TIMTRAN

Examples `:MMEMory:LOAD:IQT "C:\My Documents\Data1.iqt"`
loads IQ data from the file Data1.iqt in the My Documents folder.

:MMEMory:LOAD:STATe (No Query Form)

Loads settings from the specified file.

Syntax `:MMEMory:LOAD:STATe <file_name>`

Arguments `<file_name>::=<string>` specifies the file from which to load settings. The file extension is .cfg.

Measurement Modes All

Examples
```
:MMEMory:LOAD:STATe "C:\My Documents\Setup1.cfg"
```
loads settings from the file Setup1.cfg in the My Documents folder.

:MMEMory:LOAD:TRACe<x> (No Query Form)

Loads Trace 1 or 2 waveform data from the specified file.

Syntax `:MMEMory:LOAD:TRACe<x> <file_name>`

Arguments `<file_name>::=<string>` specifies the file from which to load trace data. The file extension is .trc.

Measurement Modes SANORMAL, SASGRAM

Examples
```
:MMEMory:LOAD:TRACe "C:\My Documents\Trace1.trc"
```
loads Trace 1 data from the file Trace1.trc in the My Documents folder.

Related Commands `:MMEMory:STORe:TRACe<x>`

:MMEMory:NAME (?)

Specifies or queries the file name when the hardcopy output destination is a file. To select the hardcopy output destination, use the :HCOPy:DESTination command.

Syntax

```
:MMEMory:NAME <file_name>
:MMEMory:NAME?
```

Arguments

<file_name>::=<string> specifies the name of the destination file. The file extension .bmp is added automatically.

Measurement Modes

All

Examples

```
:MMEMory:NAME "C:\My Documents\Screen1.bmp"
```

specifies Screen1.bmp in the My Documents folder as the destination file.

Related Commands

:HCOPy:DESTination

:MMEMory:STORe:CORRection (No Query Form)

Stores the amplitude correction table in the specified file.

Syntax

```
:MMEMory:STORe:CORRection <file_name>
```

Arguments

<file_name>::=<string> specifies the file name. The file extension is .cor.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:MMEMory:STORe:CORRection "C:\My Documents\Sample1.cor"
```

stores the amplitude correction table in the file Sample1.cor in the My Documents folder.

:MMEMory:STORe:IQT (No Query Form)

Stores IQ data in time domain in the specified file.

Syntax `:MMEMory:STORe:IQT <file_name>`

Arguments `<file_name>::=<string>` specifies the file in which to store IQ data. The file extension is .iqt.

Measurement Modes SARTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples `:MMEMory:STORe:IQT "C:\My Documents\Data1.iqt"`
stores IQ data in the file Data1.iqt in the My Documents folder.

:MMEMory:STORe:PULSe (No Query Form)

Stores the pulse measurement results in the specified file.

Syntax `:MMEMory:STORe:PULSe <file_name>`

Arguments `<file_name>::=<string>` specifies the file to store the pulse measurement results. The file extension is .csv.

Measurement Modes TIMPULSE

Examples `:MMEMory:STORe:PULSe "C:\My Documents\Result1.csv"`
stores the pulse measurement results in the Result1.csv file in the My Documents folder.

:MMEMory:STORe:STATe (No Query Form)

Stores the current settings in the specified file.

Syntax `:MMEMory:STORe:STATe <file_name>`

Arguments `<file_name>::=<string>` specifies the file in which to store the current settings. The file extension is .cfg.

Measurement Modes All

Examples `:MMEMory:STORe:STATe "C:\My Documents\Setup1.cfg"`
stores the current settings the file Setup1.cfg in the My Documents folder.

:MMEMory:STORe:TRACe<x> (No Query Form)

Stores Trace 1 or 2 waveform data in the specified file.

Syntax `:MMEMory:STORe:TRACe<x> <file_name>`

Arguments `<file_name>` specifies the file in which to store trace data.
The file extension is .trc.

Measurement Modes `SANORMAL`, `SASGRAM`

Examples `:MMEMory:STORe:TRACe1 "C:\My Documents\Trace1.trc"`
stores Trace 1 data in the file Trace1.trc in the My Documents folder.

Related Commands `:MMEMory:LOAD:TRACe<x>`

:PROGram Commands

The :PROGram commands control running a macro program.

The macro program to be run must be stored under this directory in the analyzer:

C:\Program Files\Tektronix\wca200a\Python\wca200a\measmacro

For incorporating macro programs into the analyzer, contact your local Tektronix distributor or sales office. For an example of running a macro program, refer to page 4–14.

Command Tree

Header	Parameter
:PROGram	
:CATalog?	
[:SELected]	
:DELete	
[:SELected]	
:EXECute	<command_name>
:NAME	<macro_name>
:NUMBer	<varname>,<nvalue>
:STRing	<varname>,<nvalue>

:PROGram:CATalog? (Query Only)

Queries the list of the defined macro programs.

Syntax `:PROGram:CATalog?`

Arguments None

Returns Comma-separated character strings as follows:

```
"macro_name{,macro_name}"{,"macro_name{,macro_name"}}
```

macro_name represents a macro name.

If no program has been defined, a null character (“”) is returned.

Measurement Modes All

Examples `:PROGram:CATalog?`
might return `"NONREGISTERED.MACROTEST1","NONREGISTERED.MACROTEST2"` indicating that MacroTest1 and MacroTest2 are located under the directory *C:\Program Files\Tektronix\wca200a\Python\wca200a\measmacro\nonregistered.*

:PROGram[:SELected]:DELete[:SELected] (No Query Form)

Deletes a macro program specified with the :PROGram[:SELected]:NAME command, from the memory.

Syntax `:PROGram[:SELected]:DELete[:SELected]`

Arguments None

Measurement Modes All

Examples `:PROGram:SELected:DELete:SELected`
deletes the specified macro program.

Related Commands `:PROGram[:SELected]:NAME`

:PROGram[:SELected]:EXECute (No Query Form)

Runs a command included in the macro program folder specified with the :PROGram[:SELected]:NAME command.

Syntax `:PROGram[:SELected]:EXECute <command_name>`

Arguments `<command_name>::=<string>` specifies the command.

Returns If the specified command is not found, the following error message is returned:

```
"Program Syntax error" (-285)
```

Measurement Modes All

Examples `:PROGram:SELected:EXECute "TEST1"`
runs the TEST1 command.

:PROGram[:SELected]:NAME (?)

Specifies or queries the macro program folder.

Syntax `:PROGram[:SELected]:NAME <macro_name>`

`:PROGram[:SELected]:NAME?`

Arguments `<macro_name>::=<string>` specifies the macro program folder.

Returns If the specified macro is not found, the following error message is returned:

```
"Program Syntax error" (-285)
```

Measurement Modes All

Examples `:PROGram:SELected:NAME "NONREGISTERED.MACROTEST1"`
specifies the macro program folder *MacroTest1* located under the directory *C:\Program Files\Tektronix\wca200a\Python\wca200a\measmacro\nonregistered.*

Related Commands `:PROGram[:SELected]:EXECute`

:PROGram:NUMBer (?)

Sets a numeric variable used in the macro program.

The query version of this command returns the numeric variable or the measurement result.

Syntax

```
:PROGram:NUMBer <varname>,<nvalues>
:PROGram:NUMBer? <varname>
```

Arguments

<varname>::=<string> specifies the variable.

<nvalues>::=<NRf> is the numeric value for the variable.

Returns

<NRf> is the numeric value of the specified variable.

If the specified variable is not found, the following error message is returned:

```
"Illegal variable name" (-283)
```

Measurement Modes

All

Examples

```
:PROGram:NUMBer "LOW_LIMIT",1.5
```
sets the variable LOW_LIMIT to 1.5.

```
:PROGram:NUMBer? "RESULT"
```
might return 1.2345 of the measured value stored in the variable RESULT.

:PROGram:STRing (?)

Sets a character variable used in the macro program.

The query form of this command returns the character variable or the measurement result (string).

Syntax

```
:PROGram:STRing <varname>,<svalues>
:PROGram:STRing? <varname>
```

Arguments

<varname>::=<string> specifies the variable.

<svalues>::=<string> is the string for the variable.

Returns

<string> of the specified variable.

If the specified variable is not found, the following error message is returned:

```
"Illegal variable name" (-283)
```

Measurement Modes

All

Examples

```
:PROGram:STRing "ERROR_MESSAGE","Measurement Unsuccessful"
```

sets the character string “Measurement Unsuccessful” in the variable ERROR_MESSAGE.

:READ Commands

The :READ commands acquire an input signal once in the single mode and obtain the measurement results from that data.

If you want to fetch the measurement results from the data currently residing in the memory without acquiring the input signal, use the :FETCh commands.

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Select a measurement mode with the following command:

```
:INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME
| DEMADEM | TIMCCDF | TIMTRAN | TIMPULSE }
```

2. Set the acquisition mode to single with the following command:

```
:INITiate:CONTinuous OFF
```

***NOTE**. If a :READ command is run in the continuous mode, the acquisition mode is changed to single.*

Command Tree

```
Header                  Parameter
:READ
   :ADEMod
      :AM?
         :RESult?
      :FM?
         :RESult?
      :PM?
      :PSPectrum?
   :CCDF?
   :DISTribution:CCDF?
   :OVIew?
   :PULSe?              ALL | WIDTh | PPOWer | OORatio | RIPPle
                        | PERiod | DCYCle | PHASe | CHPower
                        | OBWidth | EBWidth | FREQuency
      :SPECtrum?
      :TAMPlitude?
      :TFRequency
   :SPECtrum?
      :ACPower?
      :CFRequency?
      :CHPower?
      :CNRatio?
      :EBWidth?
      :OBWidth?
      :SPURious?
   :TRANsient
      :FVTime?
      :IQVTime?
      :PVTime?
```

:READ:ADEMod:AM? (Query Only)

Obtains the results of the AM signal analysis in time series.

Syntax `:READ:ADEMod:AM?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the chronological modulation factor data in percent (%).
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes DEMADEM

Examples `:READ:ADEMod:AM?`
might return `#41024xxxx...` (1024-byte data) for the results of the AM signal analysis.

Related Commands `:INSTrument[:SELect]`

:READ:ADEMod:AM:RESult? (Query Only)

Obtains the measurement results of the AM signal analysis.

Syntax

```
:READ:ADEMod:AM:RESult?
```

Arguments

None

Returns

```
<+AM>,<-AM>,<Total_AM>
```

Where
<+AM>::=<NRf> is the positive peak AM value in percent (%).
<-AM>::=<NRf> is the negative peak AM value in percent (%).
<Total_AM>::=<NRf> is the total AM value: (peak-peak AM value) / 2 in percent (%).

Measurement Modes

DEMADEM

Examples

```
:READ:ADEMod:AM:RESult?
```

might return `37.34,-48.75,43.04`.

Related Commands

`:INSTrument[:SELect]`

:READ:ADEMod:FM? (Query Only)

Obtains the results of the FM signal analysis in time series.

Syntax `:READ:ADEMod:FM?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the chronological frequency shift data in Hz.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes DEMADEM

Examples `:READ:ADEMod:FM?`
might return `#41024xxxx...` (1024-byte data) for the results of the FM signal analysis.

Related Commands `:INSTrument[:SELect]`

:READ:ADEMod:FM:RESult? (Query Only)

Obtains the measurement results of the FM signal analysis.

Syntax

```
:READ:ADEMod:FM:RESult?
```

Arguments

None

Returns

```
<+Pk_Freq_Dev>,<-Pk_Freq_Dev>,<P2P_Freq_Dev>,<P2P_Freq_Dev/2>,
<RMS_Freq_Dev>
```

Where
<+Pk_Freq_Dev>::=<NRf> is the positive peak frequency deviation in Hz.
<-Pk_Freq_Dev>::=<NRf> is the negative peak frequency deviation in Hz.
<P2P_Freq_Dev>::=<NRf> is the peak-to-peak frequency deviation in Hz.
<P2P_Freq_Dev/2>::=<NRf> is (peak-to-peak frequency deviation) / 2 in Hz.
<RMS_Freq_Dev>::=<NRf> is the RMS frequency deviation in Hz.

Examples

```
:READ:ADEMod:FM:RESult?
```

might return `1.13e+4,-1.55e+4,2.48e+4,1.24e+4,1.03e+4`.

Related Commands

```
:INSTrument[:SELect]
```

:READ:ADEMod:PM? (Query Only)

Obtains the results of the PM signal analysis in time series.

Syntax `:READ:ADEMod:PM?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the chronological phase shift data in degree.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes DEMADEM

Examples `:READ:ADEMod:PM?`
might return `#41024xxxx...` (1024-byte data) for the results of the PM signal analysis.

Related Commands `:INSTrument[:SELect]`

:READ:ADEMod:PSPectrum? (Query Only)

Returns spectrum data of the pulse spectrum measurement in the analog modulation analysis.

Syntax

```
:READ:ADEMod:PSPectrum?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Data(n)> is the spectrum amplitude in dBm.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 240001

Measurement Modes

DEMADEM

Examples

```
:READ:ADEMod:PSPectrum?
```

might return `#43200xxxx...` (3200-byte data) for the spectrum data.

Related Commands

```
:INSTrument[:SELect]
```

:READ:CCDF? (Query Only)

Obtains the CCDF measurement results.

Syntax `:READ:CCDF?`

Arguments None

Returns `<meanpower>,<peakpower>,<cfactor>`

Where
`<meanpower>::=<NRf>` is the average power measured value in dBm.
`<peakpower>::=<NRf>` is the peak power measured value in dBm.
`<cfactor>::=<NRf>` is the crest factor in dB.

Measurement Modes `TIMCCDF`

Examples `:READ:CCDF?`
might return `-11.16,-8.18,2.96` for the CCDF measurement results.

Related Commands `:INSTrument[:SELect]`

:READ:DISTribution:CCDF? (Query Only)

Returns the CCDF trace data in the CCDF measurement.

Syntax

```
:READ:DISTribution:CCDF?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the phase shift data in degrees for the point n.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 10001

Invalid data is returned as –1000.

Measurement Modes

TIMCCDF

Examples

```
:READ:DISTribution:CCDF?
```

might return `#41024xxxx...` (1024-byte data) for the CCDF trace data in the CCDF measurement.

Related Commands

`:READ:CCDF?`, `:INSTrument[:SELect]`

:READ:OVIew? (Query Only)

Obtains the minimum and maximum values for each 1024-point segment of waveform data displayed on the overview in the Demod (modulation analysis) and the Time (time analysis) modes.

***NOTE**. The :CONFigure:OVIew command must be run to turn measurement off before the :READ:OVIew command is executed.*

Syntax

```
:READ:OVIew?
```

Returns

```
#<Num_digit><Num_byte><MinData(1)><MaxData(1)>...
<MinData(n)><MaxData(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<MinData(n)> is the minimum data in dBm for each 1024 data point segment.
4-byte little endian floating-point format specified in IEEE 488.2
<MaxData(n)> is the maximum data in dBm for each 1024 data point segment.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 16000 (standard) / 64000 (Option 02)

Measurement Modes

DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:READ:OVIew?
```

might return `#510240xxx...` (10240-byte data) representing the minimum and the maximum values of waveform displayed on the overview.

Related Commands

:CONFigure:OVIew, :INSTrument[:SELect]

:READ:PULSe? (Query Only)

Returns the result of the pulse characteristics analysis.

Syntax

```
:READ:PULSe? { ALL | WIDTh | PPOWer | OORatio | RIPPle | PERiod
| DCYCle | PHASe | CHPower | OBWidth | EBWidth | FREQuency }
```

Arguments

Information queried is listed below for each of the arguments:

Argument	Information queried
ALL	All
WIDTh	Pulse width
PPOWer	Maximum (peak) power in the pulse-on time
OORatio	Difference between the pulse-on time power and off time power
RIPPle	Difference between the maximum and the minimum power in the pulse-on
PERiod	Time between the pulse rising edge and the next rising edge
DCYCle	Ratio of the pulse width to the pulse repetition interval (PRI)
PHASe	Phase at a certain point of each pulse
CHPower	Channel power of the pulse-on time spectrum
OBWidth	OBW (Occupied Bandwidth) of the pulse-on time spectrum
EBWidth	EBW (Emission Bandwidth) of the pulse–on time spectrum
FREQuency	Carrier frequency in the pulse-on time

Returns

Returns are listed below for each of the arguments.

ALL. <width>,<ppower>,<ooratio>,<ripple>,<period>,<dcycle>,<phase>,<chp>,<obw>,<ebw>,<freq>

Where
<width>::=<NRf> is the pulse width in s.
<ppower>::=<NRf> is the peak power in watts.
<ooratio>::=<NRf> is the on/off ratio in dB.
<ripple>::=<NRf> is the pulse ripple in watts.
<period>::=<NRf> is the pulse repetition interval in s.
<dcycle>::=<NRf> is the duty cycle in percent (%).
<phase>::=<NRf> is the pulse-pulse phase in degrees.
<chp>::=<NRf> is the channel power in watts.
<obw>::=<NRf> is the OBW in Hz.
<ebw>::=<NRf> is the EBW in Hz.
<freq>::=<NRf> is the frequency deviation in Hz.

WIDTh. #<Num_digit><Num_byte><Width(1)><Width(2)>...<Width(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Width(n)> is the pulse width value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

PPOWer. #<Num_digit><Num_byte><Ppower(1)><Ppower(2)>...
<Ppower(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Power(n)> is the peak power value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

OORatio. #<Num_digit><Num_byte><Ooratio(1)><Ooratio(2)>...
<Ooratio(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Ooratio(n)> is the on/off ratio value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

RIPPle. #<Num_digit><Num_byte><Ripple(1)><Ripple(2)>...<Ripple(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Ripple(n)> is the ripple value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

PERiod. #<Num_digit><Num_byte><Period(1)><Period(2)>...<Period(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Period(n)> is the pulse repetition interval value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

DCYCle. #<Num_digit><Num_byte><Dcycle(1)><Dcycle(2)>...<Dcycle(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Dcycle(n)> is the duty value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

PHASe. #<Num_digit><Num_byte><Phase(1)><Phase(2)>...<Phase(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Phase(n)> is the pulse–pulse phase value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

CHPower. #<Num_digit><Num_byte><Chp(1)><Chp(2)>...<Chp(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Chp(n)> is the Channel Power value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

OBWidth. #<Num_digit><Num_byte><Obw(1)><Obw(2)>...<Obw(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Obw(n)> is the OBW value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

EBWidth. #<Num_digit><Num_byte><Ebw(1)><Ebw(2)>...<Ebw(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Ebw(n)> is the EBW value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

FREQuency. #<Num_digit><Num_byte><Freq(1)><Freq(2)>...<Freq(n)>

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Freq(n)> is the frequency deviation value for each pulse number.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 1000

Measurement Modes

TIMPULSE

Examples

```
:READ:PULSe? WIDTh
```

might return #3500xxxx... (500-byte data) for the pulse width measurement result.

Related Commands

:INSTrument[:SELect]

:READ:PULSe:SPECtrum? (Query Only)

Returns the spectrum data of the frequency domain measurement in the pulse characteristics analysis.

This query command is valid when :DISPlay:PULSe:SVIew:FORMat is set to CHPowr, OBWidth, or EBWidth.

Syntax `:READ:PULSe:SPECtrum?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Data(n)> is the spectrum in dBm.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 16384

Measurement Modes TIMPULSE

Examples `:READ:PULSe:SPECtrum?`
might return `#43200xxxx...` (3200-byte data) for the spectrum data.

Related Commands `:DISPlay:PULSe:SVIew:FORMat`, `:INSTrument[:SELect]`

:READ:PULSe:TAMPlitude? (Query Only)

Returns the time domain amplitude data of the time domain measurement in the pulse characteristics analysis.

This query command is valid when :DISPlay:PULSe:SVIew:FORMat is set to WIDTh, PPOWer, OORatio, RIPPle, PERiod, DCYCle, or PHASe.

Syntax `:READ:PULSe:TAMPlitude?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
`<Num_digit>` is the number of digits in <Num_byte>.
`<Num_byte>` is the number of bytes of data that follow.
`<Data(n)>` is the absolute power for each data in watts.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 262,144

Invalid data is returned as –1000.

Measurement Modes `TIMPULSE`

Examples `:READ:PULSe:TAMPlitude?`
might return `#43200xxxx...` (3200-byte data) for the time domain amplitude.

Related Commands `:DISPlay:PULSe:SVIew:FORMat`, `:INSTrument[:SELect]`

:READ:PULSe:TFRequency? (Query Only)

Returns the frequency deviation measurement results in the pulse characteristics analysis.

This query command is valid when :DISPlay:PULSe:SVIew:FORMat is set to FREQuency.

Syntax

```
:READ:PULSe:TFRequency?
```

Arguments

None

Returns

```
#<Num_digit><Num_byre><Data(1)><Data(2)>...<Data(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of data that follow.
<Data(n)> is the frequency deviation value in Hz on the time axis.
4-byte little endian floating-point format specified IEEE 488.2.
n: Max 262,144

Invalid data is returned as –1000.

Measurement Modes

TIMPULSE

Examples

```
:READ:PULSe:TFRequency?
```

might return `#43200xxxx...` (3200-byte data) for the time domain frequency.

Related Commands

:DISPlay:PULSe:SVIew:FORMat, :INSTrument[:SELect]

:READ:SPECtrum? (Query Only)

Obtains spectrum waveform data in the S/A (spectrum analysis) mode.

Syntax `:READ:SPECtrum?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the amplitude spectrum in dBm.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 240001

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:READ:SPECtrum?`
might return `#43200xxxx...` (3200-byte data) for the spectrum waveform data.

Related Commands `:INSTrument[:SELect]`

:READ:SPECtrum:ACPower? (Query Only)

Obtains the results of the adjacent channel leakage power ratio (ACPR) measurement in the S/A mode.

Syntax

```
:READ:SPECtrum:ACPower?
```

Arguments

None

Returns

<chpower>,<acpm1>,<acpp1>,<acpm2>,<acpp2>,<acpm3>,<acpp3>

Where
<chpower>::=<NRf> is the channel power measured value in dBm.
<acpm1>::=<NRf> is the first lower adjacent channel ACPR in dB.
<acpp1>::=<NRf> is the first upper adjacent channel ACPR in dB.
<acpm2>::=<NRf> is the second lower adjacent channel ACPR in dB.
<acpp2>::=<NRf> is the second upper adjacent channel ACPR in dB.
<acpm3>::=<NRf> is the third lower adjacent channel ACPR in dB.
<acpp3>::=<NRf> is the third upper adjacent channel ACPR in dB.

***NOTE**. All the values may not be returned when the adjacent channel(s) goes out of the span due to the settings of the channel bandwidth and spacing (refer to the [:SENSe]:ACPower subgroup). For example, if the third adjacent channel goes out of the span, the response is <chpower>,<acpm1>,<acpp1>,<acpm2>, <acpp2>; <acpm3> and <acpp3> are not returned.*

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:READ:SPECtrum:ACPower?
```

might return -11.38,-59.41,-59.51,-59.18,-59.31,-59.17,-59.74 for the ACPR measurement results.

Related Commands

:INSTrument[:SELect], [:SENSe]:ACPower subgroup

:READ:SPECtrum:CFRequency? (Query Only)

Obtains the results of the carrier frequency measurement in the S/A mode.

Syntax `:READ:SPECtrum:CFRequency?`

Arguments None

Returns `<cfreq>::=<NRf>` is the measured value of the carrier frequency in Hz.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples
```
:READ:SPECtrum:CFRequency?
```
might return `846187328.5` for the carrier frequency.

Related Commands `:INSTrument[:SELect]`

:READ:SPECtrum:CHPower? (Query Only)

Obtains the results of the channel power measurement in the S/A mode.

Syntax `:READ:SPECtrum:CHPower?`

Arguments None

Returns `<chpower>::=<NRf>` is the channel power measured value in dBm.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples
```
:READ:SPECtrum:CHPower?
```
might return `-1.081` for the measurement results of the channel power.

Related Commands `:INSTrument[:SELect]`

:READ:SPECtrum:CNRatio? (Query Only)

Obtains the results of the carrier-to-noise ratio (C/N) measurement in the S/A (spectrum analysis) mode.

Syntax

```
:READ:SPECtrum:CNRatio?
```

Arguments

None

Returns

```
<ctn>,<ctno>
```

Where
<ctn>::=<NRf> is the measured value of C/N in dB.
<ctno>::=<NRf> is the measured value of C/No in dB/Hz.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:READ:SPECtrum:CNRatio?
```

might return 75.594,125.594 for the C/N measurement results.

Related Commands

:INSTrument[:SELect]

:READ:SPECtrum:EBWidth? (Query Only)

Obtains the results of the emission bandwidth (EBW) measurement in the S/A (spectrum analysis) mode.

Syntax `:READ:SPECtrum:EBWidth?`

Arguments None

Returns `<ebw>::=<NRf>` is the measured value of EBW in Hz.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:READ:SPECtrum:EBWidth?`
might return `30956.26` for the EBW measurement results.

Related Commands `:INSTrument[:SELect]`

:READ:SPECtrum:OBWidth? (Query Only)

Obtains the results of the occupied bandwidth (OBW) measurement in the S/A (spectrum analysis) mode.

Syntax `:READ:SPECtrum:OBWidth?`

Arguments None

Returns `<obw>::=<NRf>` is the measured value of OBW in Hz.

Measurement Modes SANORMAL, SASGRAM, SARTIME

Examples `:READ:SPECtrum:OBWidth?`
might return `26510.163` for the OBW measurement results.

Related Commands `:INSTrument[:SELect]`

:READ:SPECtrum:SPURious? (Query Only)

Obtains the results of the spurious signal measurement in the S/A (spectrum analysis) mode.

Syntax

```
:READ:SPECtrum:SPURious?
```

Arguments

None

Returns

```
<snum>{,<dfreq>,<rdb>}
```

Where
<snum>::=<NR1> is the number of detected spurious emissions, max. 20
<dfreq>::=<NRf> is the detuned frequency of spurious relative to carrier in Hz.
<rdb>::=<NRf> is the relative level of spurious signal to carrier in dB.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:READ:SPECtrum:SPURious?
```

might return `3,1.2E6,-79,2.4E6,-79.59,1E6,-80.38` for the spurious signal measurement.

Related Commands

```
:INSTrument[:SELect]
```

:READ:TRANsient:FVTime? (Query Only)

Obtains the results of the frequency vs. time measurement in the Time (time analysis) mode.

Syntax `:READ:TRANsient:FVTime?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the chronological frequency data in Hz.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes `TIMTRAN`

Examples `:READ:TRANsient:FVTime?`
might return `#41024xxxx...` (1024-byte data) for the results of the frequency vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:READ:TRANsient:IQVTime? (Query Only)

Obtains the results of the IQ level vs. time measurement in the Time (time analysis) mode.

Syntax

```
:READ:TRANsient:IQVTime?
```

Arguments

None

Returns

```
#<Num_digit><Num_byte><Idata(1)><Qdata(1)>
<Idata(2)><Qdata2>...<Idata(n)><Qdata(n)>
```

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Idata(n)><Qdata(n)> is the I and Q signal level data in volt.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes

TIMTRAN

Examples

```
:READ:TRANsient:IQVTime?
```

might return #41024xxxx... (1024-byte data) for the results of the IQ level vs. time measurement.

Related Commands

:INSTrument[:SELect]

:READ:TRANsient:PVTime? (Query Only)

Obtains the results of the power measurement vs. time in the Time (time analysis) mode.

Syntax `:READ:TRANsient:PVTime?`

Arguments None

Returns `#<Num_digit><Num_byte><Data(1)><Data(2)>...<Data(n)>`

Where
<Num_digit> is the number of digits in <Num_byte>.
<Num_byte> is the number of bytes of the data that follow.
<Data(n)> is the chronological power data in dBm.
4-byte little endian floating-point format specified in IEEE 488.2
n: Max 512000 (= 1024 points × 500 frames)

Measurement Modes TIMTRAN

Examples `:READ:TRANsient:PVTime?`
might return `#41024xxxx...` (1024-byte data) for the results of the power vs. time measurement.

Related Commands `:INSTrument[:SELect]`

:SENSe Commands

The :SENSe commands set the details for each of the measurement sessions. They are divided into the following subgroups:

Table 2–36: :SENSe command subgroups

Command header	Function	Refer to:
[:SENSE]:ACPower	Set up ACPR measurement	p. 2–236
[:SENSE]:ADEMod	Set up analog modulation analysis	p. 2–240
[:SENSE]:AVERage	Set up average	p. 2–246
[:SENSE]:BSIZe	Set the block size	p. 2–249
[:SENSE]:CCDF	Set up CCDF measurement	p. 2–250
[:SENSE]:CFRequency	Set up carrier frequency measurement	p. 2–253
[:SENSE]:CHPower	Set up channel power measurement	p. 2–254
[:SENSE]:CNRatio	Set up C/N measurement	p. 2–257
[:SENSE]:CORRection	Set up amplitude correction	p. 2–262
[:SENSE]:EBWidth	Set up EBW measurement	p. 2–267
[:SENSE]:FEED	Set up signal path	p. 2–269
[:SENSE]:FREQuency	Set up frequency-related conditions	p. 2–270
[:SENSE]:OBWidth	Set up OBW measurement	p. 2–279
[:SENSE]:PULSe	Set up pulse characteristics measurement.	p. 2–281
[:SENSE]:ROSCillator	Set up reference oscillator	p. 2–290
[:SENSE]:SPECtrum	Set up spectrum measurement	p. 2–291
[:SENSE]:SPURious	Set up spurious signal measurement	p. 2–306
[:SENSE]:TRANsient	Set up time domain measurement	p. 2–310

[:SENSe]:ACPower Subgroup

The [:SENSe]:ACPower commands set up the conditions related to the adjacent channel leakage power ratio (ACPR) measurement in the S/A (spectrum analysis) mode.

Command Tree

Header	Parameter
[SENSe]	
:ACPower	
:BANDwidth|:BWIDth	
:ACHannel	<frequency>
:INTegration	<frequency>
:CSPacing	<frequency>
:FILTer	
:COEFficient	<numeric_value>
:TYPE	RECTangle | GAUSsian | NYQuist | RNYQuist

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   :INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME }
   ```

2. Run one of the following commands to start the ACPR measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:ACPower`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement ACPower`

[:SENSe]:ACPower:BANDwidth|:BWIDth:ACHannel(?)

Sets or queries the bandwidth of the adjacent channels for the ACPR measurement (see Figure 2–15).

Syntax

```
[:SENSe]:ACPower:BANDwidth|:BWIDth:ACHannel <value>
[:SENSe]:ACPower:BANDwidth|:BWIDth:ACHannel?
```

Arguments

<value>::=<NRf> specifies the bandwidth of the adjacent channels for the ACPR measurement. Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:ACPower:BANDwidth:ACHannel 3.5MHz
```
sets the bandwidth of the adjacent channels to 3.5 MHz.

[:SENSe]:ACPower:BANDwidth|:BWIDth:INTegration(?)

Sets or queries the bandwidth of the main channel for the ACPR measurement (see Figure 2–15).

Syntax

```
[:SENSe]:ACPower:BANDwidth|:BWIDth:INTegration <value>
[:SENSe]:ACPower:BANDwidth|:BWIDth:INTegration?
```

Arguments

<value>::=<NRf> specifies the bandwidth of the main channel for the ACPR measurement. Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:ACPower:BANDwidth:INTegration 3.5MHz
```
sets the bandwidth of the main channel to 3.5 MHz.

[:SENSe]:ACPower:CSPacing(?)

Sets or queries the channel-to-channel spacing for the ACPR measurement (see Figure 2–15).

Syntax

```
[:SENSe]:ACPower:CSPacing <value>
[:SENSe]:ACPower:CSPacing?
```

Arguments

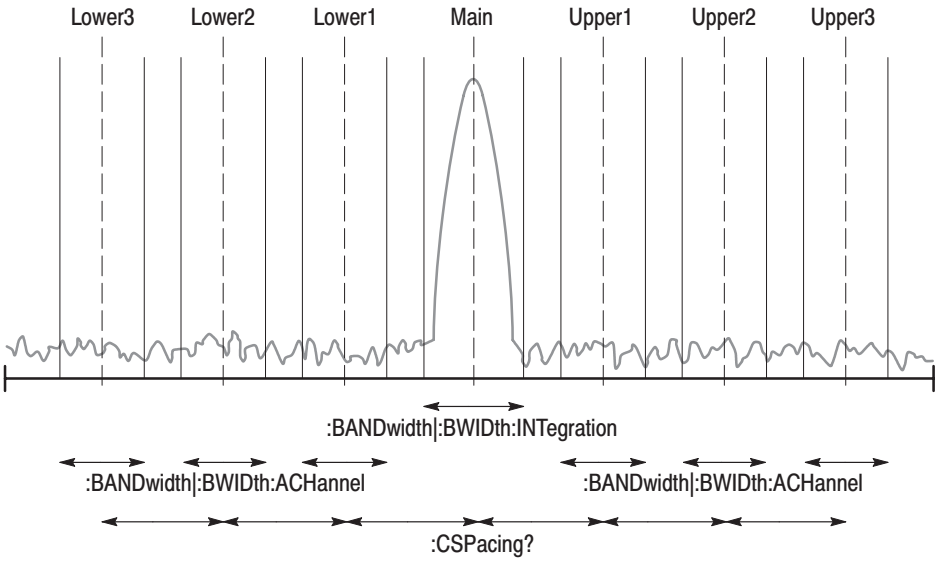
<value>::=<NRf> specifies the channel-to-channel spacing for the ACPR measurement. Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:ACPower:CSPacing 5MHz
```
sets the channel-to-channel spacing to 5 MHz.



NOTE: The command header [:SENSe]:ACPower is omitted here.

Figure 2–15: Setting up the ACPR measurement

[:SENSe]:ACPower:FILTer:COEFficient(?)

Sets or queries the filter roll-off rate for the ACPR measurement when you have selected either NYQuist (Nyquist filter) or RNYQuist (Root Nyquist filter) in the [:SENSe]:ACPower:FILTer:TYPE command.

Syntax

```
[:SENSe]:ACPower:FILTer:COEFficient <ratio>
[:SENSe]:ACPower:FILTer:COEFficient?
```

Arguments

`<ratio>::=<NRf>` specifies the roll-off rate. Range: 0 to 1.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:ACPower:FILTer:COEFficient 0.5
```
sets the filter roll-off rate to 0.5.

Related Commands

`[:SENSe]:ACPower:FILTer:TYPE`

[:SENSe]:ACPower:FILTer:TYPE(?)

Selects or queries the filter for the ACPR measurement.

Syntax

```
[:SENSe]:ACPower:FILTer:TYPE { RECTangle | GAUSsian | NYQuist
| RNYQuist }
[:SENSe]:ACPower:FILTer:TYPE?
```

Arguments

`RECTangle` selects the rectangular filter.

`GAUSsian` selects the Gaussian filter.

`NYQuist` selects the Nyquist filter (default).

`RNYQuist` selects the Root Nyquist filter.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:ACPower:FILTer:TYPE NYQuist
```
selects the Nyquist filter for the ACPR measurement.

[:SENSe]:ADEMod Subgroup

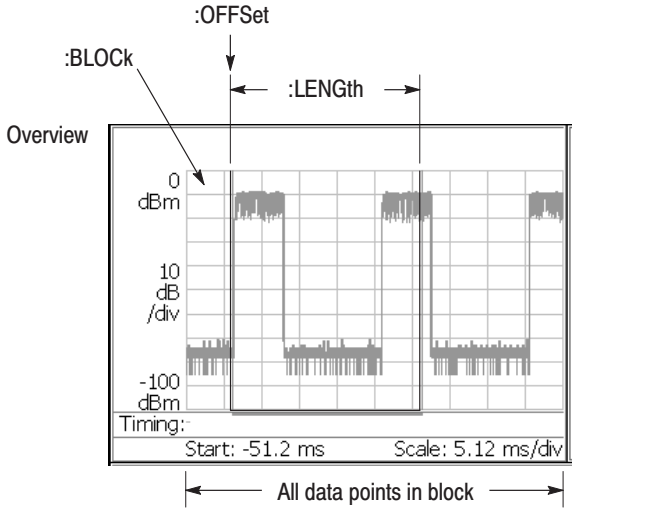
Sets up the analog modulation analysis.

***NOTE**. To use a command of this group, you must have selected DEMADEM (analog modulation analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
[:SENSe]	
:ADEMod	
:BLOCk	<numeric_value>
:CARRier	
:OFFSet	<numeric_value>
:SEARch	<boolean>
:FM	
:THReshold	<numeric_value>
[:IMMediate]	
:LENGth	<numeric_value>
:MODulation	AM | FM | PM | IQVT | OFF
:OFFSet	<numeric_value>
:PM	
:THReshold	<numeric_value>

For the commands defining the analysis range, see the figure below. The analysis range is shown by a green line in the overview.



NOTE: Command header [:SENSe]:ADEMod is omitted here.

Figure 2–16: Defining the analysis range

[:SENSe]:ADEMod:BLOCk(?)

Sets or queries the number of the block to measure in the analog modulation analysis (see Figure 2–16).

Syntax

```
[:SENSe]:ADEMod:BLOCk <number>
[:SENSe]:ADEMod:BLOCk?
```

Arguments

<number>::=<NR1> specifies the block number. Zero represents the latest block. Range: –M to 0 (M: Number of acquired blocks)

Measurement Modes

DEMADEM

Examples

```
:SENSe:ADEMod:BLOCk -5
```

sets the block number to –5.

[:SENSe]:ADEMod:CARRier:OFFSet(?)

Sets or queries the carrier frequency offset in the FM signal analysis.

Syntax

```
[:SENSe]:ADEMod:CARRier:OFFSet <freq>
[:SENSe]:ADEMod:CARRier:OFFSet?
```

Arguments

<freq>::=<NR1> is the carrier frequency offset. Range: –30 to +30 MHz

Measurement Modes

DEMADEM

Examples

```
:SENSe:ADEMod:CARRier:OFFSet 10MHz
```

sets the carrier frequency offset to 10 MHz.

Related Commands

[:SENSe]:ADEMod:CARRier:SEARch

[:SENSe]:ADEMod:CARRier:SEARch(?)

Determines whether to detect the carrier automatically in the FM signal analysis.

Syntax

```
[:SENSe]:ADEMod:CARRier:SEARch  { 0 | 1 | OFF | ON }
[:SENSe]:ADEMod:CARRier:SEARch?
```

Arguments

`OFF` or `0` specifies that the carrier is not detected automatically. To set it, use the [:SENSe]:ADEMod:CARRier:OFFSet command.

`ON` or `1` specifies that the carrier is detected automatically.

Measurement Modes

`DEMADEM`

Examples

```
:SENSe:ADEMod:CARRier:SEARch ON
```

specifies that the carrier is detected automatically.

Related Commands

`[:SENSe]:ADEMod:CARRier:OFFSet`

[:SENSe]:ADEMod:FM:THReshold(?)

Sets or queries the threshold level above which the input signal is determined to be a burst in the FM signal analysis. The burst detected first is used for the measurement.

Syntax

```
[:SENSe]:ADEMod:FM:THReshold <value>
[:SENSe]:ADEMod:FM:THReshold?
```

Arguments

`<value>::=<NRf>` specifies the threshold level. Range: –100.0 to 0.0 dB.

Measurement Modes

`DEMADEM`

Examples

```
:SENSe:ADEMod:FM:THReshold -10
```

sets the threshold level to –10 dB.

[:SENSe]:ADEMod[:IMMediate] (No Query Form)

Runs the analog demodulation calculation for the acquired data. To select the analog demodulation method, use the [:SENSe]:ADEMod:MODulation command. To acquire data, use the :INITiate command.

Syntax `[:SENSe]:ADEMod[:IMMediate]`

Arguments None

Measurement Modes `DEMADEM`

Examples `:SENSe:ADEMod:IMMediate`
runs the analog demodulation calculation.

Related Commands `:INITiate`, `[:SENSe]:ADEMod:MODulation`

[:SENSe]:ADEMod:LENGth(?)

Sets or queries the range for the analog modulation analysis (see Figure 2–16).

Syntax `[:SENSe]:ADEMod:LENGth <value>`

`[:SENSe]:ADEMod:LENGth?`

Arguments `<value>::=<NR1>` specifies the analysis range by the number of data points. Range: 1 to 1024 × (Block size). To set the block size, use the [:SENSe]:BSIZe command.

Measurement Modes `DEMADEM`

Examples `:SENSe:ADEMod:LENGth 1000`
sets the analysis range to 1000 points.

Related Commands `[:SENSe]:BSIZe`

[:SENSe]:ADEMod:MODulation(?)

Selects or queries the measurement item of the analog modulation analysis.

Syntax

```
[:SENSe]:ADEMod:MODulation { AM | FM | PM | IQVT | OFF }
[:SENSe]:ADEMod:MODulation?
```

Arguments

The arguments and measurement items are listed below:

Table 2–37: Measurement item selections

Argument	Measurement item
AM	AM signal analysis
FM	FM signal analysis
PM	PM signal analysis
IQVT	IQ level vs. time measurement
OFF	Turns off the measurement.

Measurement Modes

DEMADEM

Examples

```
:SENSe:ADEMod:MODulation PM
```

selects the PM signal analysis.

[:SENSe]:ADEMod:OFFSet(?)

Sets or queries the measurement start position for the analog modulation analysis (see Figure 2–16).

Syntax

```
[:SENSe]:ADEMod:OFFSet <value>
[:SENSe]:ADEMod:OFFSet?
```

Arguments

<value>::=<NR1> specifies the measurement start position by the number of points. Range: 0 to 1024 × (Block size) – 1. To set the block size, use the [:SENSe]:BSIZe command.

Measurement Modes

DEMADEM

Examples

```
:SENSe:ADEMod:OFFSet 500
```

sets the measurement start position to point 500.

Related Commands

[:SENSe]:BSIZe

[:SENSe]:ADEMod:PM:THReshold(?)

Sets or queries the threshold level above which the input signal is determined to be a burst in the PM signal analysis. The burst detected first is used for the measurement.

Syntax

```
[:SENSe]:ADEMod:PM:THReshold <value>
[:SENSe]:ADEMod:PM:THReshold?
```

Arguments

<value>::=<NRf> specifies the threshold level. Range: –100.0 to 0.0 dB.

Measurement Modes

DEMADEM

Examples

```
:SENSe:ADEMod:PM:THReshold -10
```

sets the threshold level to –10 dB.

[:SENSe]:AVERage Subgroup

The [:SENSe]:AVERage commands control average process for measured values in the modulation analysis (Demod mode) and the time analysis (Time mode).

***NOTE**. In fact, data is always acquired without averaging in the Demod and the Time modes.*

Command Tree

Header	Parameter
[:SENSe]	
:AVERage	
:CLEar	
:COUNt	<numeric_value>
[:STATE]	<boolean>
:TCONtrol	EXPonential | REPeat

[:SENSe]:AVERage:CLEar (No Query Form)

Clears average data and counter, and restarts the average process.

Syntax `[:SENSe]:AVERage:CLEar`

Arguments None

Measurement Modes `DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples `:SENSe:AVERage:CLEar`
clears average data and counter, and restarts the average process.

[:SENSe]:AVERage:COUNt(?)

Sets or queries the number of traces to combine using the RMS average. After :COUNt traces have been averaged, the average process is controlled by the :TCONtrol setting (refer to page 2–248).

Syntax `[:SENSe]:AVERage:COUNt <value>`

`[:SENSe]:AVERage:COUNt?`

Arguments `<value>::=<NR1>` is the number of traces to combine for averaging. Range: 1 to 100000 (default: 20)

Measurement Modes `DEMADEM`, `TIMTRAN`, `TIMPULSE`

Examples `:SENSe:AVERage:COUNt 64`
sets the average count to 64.

Related Commands `[:SENSe]:AVERage:TCONtrol`

[:SENSe]:AVERage[:STATe](?)

Determines whether to turn averaging on or off.

Syntax

```
[:SENSe]:AVERage[:STATe] { OFF | ON | 0 | 1 }
[:SENSe]:AVERage[:STATe]?
```

Arguments

OFF or 0 turns off averaging.

ON or 1 turns on averaging.

Measurement Modes

DEMADEM, TIMTRAN, TIMPULSE

Examples

```
:SENSe:AVERage:STATe ON
```

turns on averaging.

[:SENSe]:AVERage:TCONtrol(?)

Selects or queries the action when more than :AVERage:COUNt measurement results are generated (TCONtrol is TerminalCONtrol).

Syntax

```
[:SENSe]:AVERage:TCONtrol { EXPonential | REPeat }
[:SENSe]:AVERage:TCONtrol?
```

Arguments

EXPonential continues the RMS (root-mean-square) average with an exponential weighting applied to old values using the setting of [:SENSe]:AVERage:COUNt as the weighting factor.

REPeat clears average data and counter, and restarts the average process when :AVERage:COUNt is reached.

Measurement Modes

DEMADEM, TIMTRAN, TIMPULSE

Examples

```
:SENSe:AVERage:TCONtrol REPeat
```

repeats the averaging process.

Related Commands

[:SENSe]:AVERage:COUNt, [:SENSe]:AVERage:TYPE

[:SENSe]:BSIZe Subgroup

The [:SENSe]:BSIZe command controls the block size (the number of frames in each contiguous acquisition).

***NOTE**. This subgroup is available in the Real Time S/A (real-time spectrum analysis), the Demod (modulation analysis), and the Time (time analysis) modes.*

Command Tree

Header	Parameter
[:SENSe]	
:BSIZe	<numeric_value>

[:SENSe]:BSIZe(?)

Sets or queries the block size.

Syntax

```
[:SENSe]:BSIZe <value>
[:SENSe]:BSIZe?
```

Arguments

<value>::=<NR1> specifies the block size. The range depends on the trigger mode set with the :TRIGger[:SEQuence]:MODE command as shown in Table 2–38.

Table 2–38: Block size setting range

Trigger mode	Block size
AUTO	1 to 500
NORMal	5 to 500

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:SENSe:BSIZe 8
```

sets the block size to 8.

Related Commands

:TRIGger[:SEQuence]:MODE

[:SENSe]:CCDF Subgroup

The [:SENSe]:CCDF commands set up the conditions related to the CCDF measurement.

***NOTE**. To use a command of this group, you must have selected TIMCCDF (CCDF measurement) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
[:SENSe]	
:CCDF	
:BLOCk	<numeric_value>
:CLEar	
:RMEasurement	
:THReshold	<numeric_value>

[:SENSe]:CCDF:BLOCk(?)

Sets or queries the number of the block to measure in the CCDF analysis.

Syntax

```
[:SENSe]:CCDF:BLOCk <value>
[:SENSe]:CCDF:BLOCk?
```

Arguments

<value>::=<NR1> specifies the block number. Zero represents the latest block. Range: –M to 0 (M: Number of acquired blocks)

Measurement Modes

TIMCCDF

Examples

```
:SENSe:CCDF:BLOCk -5
```

sets the block number to –5.

[:SENSe]:CCDF:CLEar (No Query Form)

Resets the CCDF measurement.

Syntax

```
[:SENSe]:CCDF:CLEar
```

Arguments

None

Measurement Modes

TIMCCDF

Examples

```
:SENSe:CCDF:CLEar
```

resets the CCDF measurement.

[:SENSe]:CCDF:RMEasurement (No Query Form)

Clears the CCDF accumulator and restarts the measurement.
This command is equivalent to the [:SENSe]:CCDF:CLEar command.

Syntax `[:SENSe]:CCDF:RMEasurement`

Arguments None

Measurement Modes `TIMCCDF`

Examples `:SENSe:CCDF:RMEasurement`
clears the CCDF accumulator and restarts the measurement.

Related Commands `[:SENSe]:CCDF:CLEar`

[:SENSe]:CCDF:THReshold(?)

Sets or queries the threshold which defines the samples to be included in the CCDF calculation.

Syntax `[:SENSe]:CCDF:THReshold <value>`

`[:SENSe]:CCDF:THReshold?`

Arguments `<value>::=<NR1>` specifies the threshold. Range: –250 to 130 dBm.

Measurement Modes `TIMCCDF`

Examples `:SENSe:CCDF:THReshold 50dBm`
sets the threshold to 50 dBm.

[:SENSe]:CFRequency Subgroup

The [:SENSe]:CFRequency commands set up the conditions related to the carrier frequency measurement.

Command Tree

Header	Parameter
[:SENSe]	
:CFRequency	
:CRESolution	<numeric_value>

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME }
   ```

2. Run one of the following commands to start the carrier frequency measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:CFRequency`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement CFRequency`

[:SENSe]:CFRequency:CRESolution(?)

Sets or queries the counter resolution for the carrier frequency measurement.

Syntax

```
[:SENSe]:CFRequnecy:CRESolution <value>
[:SENSe]:CFRequnecy:CRESolution?
```

Arguments

<value>::=<NRf> specifies the counter resolution.
Setting value (Hz): 0.001, 0.01, 0.1, 1, 10, 100, 1k, 10k, 100k, or 1M

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CFRequnecy:CRESolution 1kHz
```
sets the counter resolution to 1 kHz.

[:SENSe]:CHPower Subgroup

The [:SENSe]:CHPower commands set up the conditions related to the channel power measurement.

Command Tree

Header	Parameter
[:SENSe]	
:CHPower	
:BANDwidth|:BWIDth	
:INTegration	<numeric_value>
:FILTer	
:COEFficient	<numeric_value>
:TYPE	RECTangle | GAUSsian | NYQuist | RNYQuist

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME }
   ```

2. Run one of the following commands to start the channel power measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:CHPower`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement CHPower`

[:SENSe]:CHPower:BANDwidth|:BWIDth:INTegration(?)

Sets or queries the channel bandwidth for the channel power measurement (see Figure 2–17).

Syntax

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

Arguments

`<value>::=<NRf>` specifies the channel bandwidth for the channel power measurement. Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

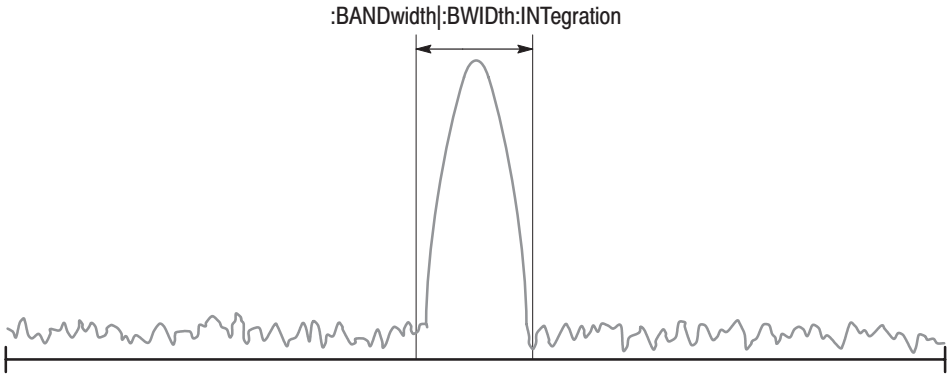
Measurement Modes

`SANORMAL`, `SASGRAM`, `SARTIME`

Examples

```
:SENSe:CHPower:BANDwidth:INTegration 2.5MHz
```

sets the channel bandwidth to 2.5 MHz.



NOTE: Command header [:SENSe]:CHPower is omitted here.

Figure 2–17: Setting up the channel power measurement

[:SENSe]:CHPower:FILTer:COEFficient(?)

Sets or queries the roll-off rate of the filter for the channel power measurement when you have selected either NYQuist (Nyquist filter) or RNYQuist (Root Nyquist filter) in the [:SENSe]:CHPower:FILTer:TYPE command.

Syntax

```
[:SENSe]:CHPower:FILTer:COEFficient <ratio>
[:SENSe]:CHPower:FILTer:COEFficient?
```

Arguments

<ratio>::=<NRf> specifies the roll-off rate of the filter for the channel power measurement. Range: 0.0001 to 1 (default: 0.5)

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CHPower:FILTer:COEFficient 0.3
```
sets the filter roll-off rate to 0.3.

Related Commands

[:SENSe]:CHPower:FILTer:TYPE

[:SENSe]:CHPower:FILTer:TYPE(?)

Selects or queries the filter for the channel power measurement.

Syntax

```
[:SENSe]:CHPower:FILTer:TYPE { RECTangle | GAUSsian | NYQuist
| RNYQuist }
[:SENSe]:CHPower:FILTer:TYPE?
```

Arguments

RECTangle selects the rectangular filter.

GAUSsian selects the Gaussian filter.

NYQuist selects the Nyquist filter (default).

RNYQuist selects the Root Nyquist filter.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CHPower:FILTer:TYPE RNYQuist
```
selects the Root Nyquist filter.

[:SENSe]:CNRatio Subgroup

The [:SENSe]:CNRatio commands set up the conditions related to the carrier-to-noise ratio (C/N) measurement.

Command Tree

Header	Parameter
[:SENSe]	
:CNRatio	
:BANDwidth|:BWIDth	
:INTegration	<frequency>
:NOISe	<frequency>
:FILTer	
:COEFficient	<numeric_value>
:TYPE	RECTangle | GAUSsian | NYQuist | RNYQuist
:OFFSet	<frequency>

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   :INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME }
   ```

2. Run one of the following commands to start the C/N measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:CNRatio`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement CNRatio`

[:SENSe]:CNRatio:BANDwidth|:BWIDth:INTegration(?)

Sets or queries the channel bandwidth for the C/N measurement (see Figure 2–18).

Syntax

```
[:SENSe]:CNRatio:BANDwidth|:BWIDth:INTegration <value>
[:SENSe]:CNRatio:BANDwidth|:BWIDth:INTegration?
```

Arguments

<value>::=<NRf> is the carrier bandwidth for the C/N measurement.
Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

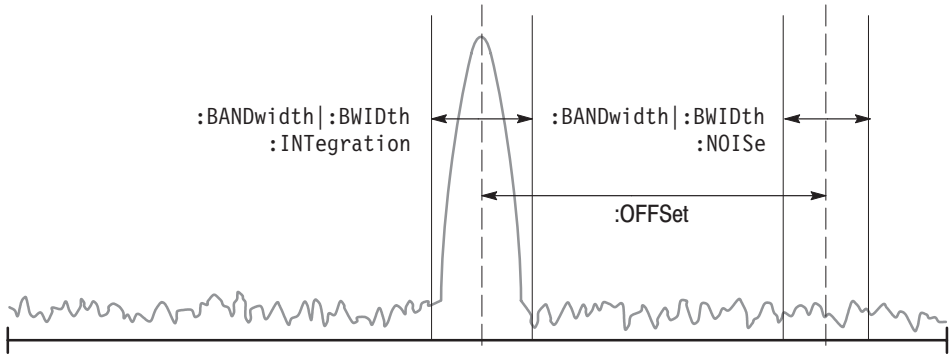
Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CNRatio:BANDwidth:INTegration 1MHz
```

sets the carrier bandwidth to 1 MHz.



NOTE: Command header [:SENSe]:CNRatio is omitted here.

Figure 2–18: Setting up the C/N measurement

[:SENSe]:CNRatio:BANDwidth|:BWIDth:NOISe(?)

Sets or queries the noise bandwidth for the C/N measurement (see Figure 2–18).

Syntax

```
[:SENSe]:CNRatio:BANDwidth|:BWIDth:NOISe <value>
[:SENSe]:CNRatio:BANDwidth|:BWIDth:NOISe?
```

Arguments

<value>::=<NRf> is the noise bandwidth for the C/N measurement.
Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CNRatio:BANDwidth:NOISe 1.5MHz
```
sets the noise bandwidth to 1.5 MHz.

[:SENSe]:CNRatio:FILTer:COEFficient(?)

Sets or queries the roll-off rate of the filter for the C/N measurement when you have selected either NYQuist (Nyquist filter) or RNYQuist (Root Nyquist filter) in the [:SENSe]:CNRatio:FILTer:TYPE command.

Syntax

```
[:SENSe]:CNRatio:FILTer:COEFficient <value>
[:SENSe]:CNRatio:FILTer:COEFficient?
```

Arguments

<value>::=<NRf> is the filter roll-off rate. Range: 0.0001 to 1 (default: 0.5)

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CNRatio:FILTer:COEFficient 0.3
```
sets the filter roll-off rate to 0.3.

Related Commands

[:SENSe]:CNRatio:FILTer:TYPE

[:SENSe]:CNRatio:FILTer:TYPE(?)

Selects or queries the filter for the C/N measurement.

Syntax

```
[:SENSe]:CNRatio:FILTer:TYPE { RECTangle | GAUSsian | NYQuist | RNYQuist }
[:SENSe]:CNRatio:FILTer:TYPE?
```

Arguments

RECTangle selects the rectangular filter.

GAUSsian selects the Gaussian filter.

NYQuist selects the Nyquist filter (default).

RNYQuist selects the Root Nyquist filter.

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:CNRatio:FILTer:TYPE RNYQuist
```
selects the Root Nyquist filter.

[:SENSe]:CNRatio:OFFSet(?)

Sets or queries offset from the carrier to noise in the the C/N measurement (see Figure 2–18).

Syntax

`[:SENSe]:CNRatio:OFFSet <freq>`

`[:SENSe]:CNRatio:OFFSet?`

Arguments

`<freq>::=<NRf>` specifies the offset frequency. Range: –(Span)/2 to +(Span)/2

Measurement Modes

`SANORMAL`, `SASGRAM`, `SARTIME`

Examples

```
:SENSe:CNRatio:OFFSet 5MHz
```

sets the offset frequency to 5 MHz.

[:SENSe]:CORRection Subgroup

The [:SENSe]:CORRection commands control the amplitude correction. For details on the amplitude correction, refer to the *RSA2203A and RSA2208A User Manual*.

***NOTE**. This subgroup is available in the S/A (spectrum analysis) mode except real-time. You must have selected SANORMAL or SASGRAM with the :INSTrument[:SELect] command to use a command in this subgroup but only [:SENSe]:CORRection[:MAGNitude] command which is available in all the measurement modes.*

Command Tree

Header	Parameter
[:SENSe]	
:CORRection	
:DATA	#<Num_digit><Num_byte> <Freq(1)><Ampl(1)> <Freq(2)><Ampl(2)>... <Freq(n)><Ampl(n)>
:DELete	
:OFFSet	
[:MAGNitude]	<numeric_value>
:FREQency	<numeric_value>
[:STATe]	
:X	
:SPACing	LINear | LOGarithmic
:Y	
:SPACing	LINear | LOGarithmic

[:SENSe]:CORRection:DATA(?)

Sets or queries the amplitude correction data.

Syntax

```
[:SENSe]:CORRection:DATA #<Num_digit><Num_byte>
   <Freq(1)><Ampl(1)><Freq(2)><Ampl(2)>...<Freq(n)><Ampl(n)>
```

```
[:SENSe]:CORRection:DATA?
```

Arguments

<Num_digit> is the number of digits in <Num_byte>.

<Num_byte> is the number of bytes of the data that follow.

<Freq(n)> is the frequency at correction point in Hz.
4-byte little endian floating–point format specified in IEEE 488.2

<Ampl(n)> is the amplitude correction value at frequency <Freq(n)> in dB.
4-byte little endian floating-point format specified in IEEE 488.2

Enter the data that consists of pairs of the frequency and amplitude correction values (n: Max 3000).

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:CORRection:DATA #41024xxxx...
```
sets the correction values at 1024 points.

[:SENSe]:CORRection:DELete (No Query Form)

Deletes all the amplitude correction data.

Syntax

```
[:SENSe]:CORRection:DELete
```

Arguments

None

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:CORRection:DELete
```
deletes all the amplitude correction data.

[:SENSe]:CORRection:OFFSet[:MAGNitude](?)

Sets or queries the amplitude offset value in the amplitude correction.

Syntax

```
[:SENSe]:CORRection:OFFSet[:MAGNitude] <value>
[:SENSe]:CORRection:OFFSet[:MAGNitude]?
```

Arguments

<value>::=<NRf> specifies the amplitude offset value.
Range: –200 to +200 dB.

Measurement Modes

All

Examples

```
:SENSe:CORRection:OFFSet:MAGNitude 10
```
sets the amplitude offset value to 10 dB.

Related Commands

[:SENSe]:CORRection:OFFSet:STATe

[:SENSe]:CORRection:OFFSet:FREQuency(?)

Sets or queries the frequency offset value in the amplitude correction.

Syntax

```
[:SENSe]:CORRection:OFFSet:FREQuency <value>
[:SENSe]:CORRection:OFFSet:FREQuency?
```

Arguments

<value>::=<NRf> specifies the frequency offset value.
Range: –100 GHz to +100 GHz.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:CORRection:OFFSet:FREQuency 10MHz
```
sets the frequency offset value to 10 MHz.

Related Commands

[:SENSe]:CORRection:OFFSet:STATe

[:SENSe]:CORRection[:STATe](?)

Determines whether to turn the amplitude correction on or off.

Syntax

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

Arguments

OFF or 0 turns off the amplitude correction.

ON or 1 turns on the amplitude correction.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:CORRection:STATe ON
```

turns on the amplitude correction.

[:SENSe]:CORRection:X:SPACing(?)

Determines whether the horizontal, or frequency, scaling is linear or logarithmic for interpolation of amplitude correction data.

Syntax

```
[:SENSe]:CORRection:X:SPACing { LINear | LOGarithmic }
[:SENSe]:CORRection:X:SPACing?
```

Arguments

`LINear` selects the linear scale for the interpolation.

`LOGarithmic` selects the logarithmic scale for the interpolation.

Measurement Modes

`SANORMAL`, `SASGRAM`

Examples

```
:SENSe:CORRection:X:SPACing LINear
```
selects the linear scale for the interpolation.

[:SENSe]:CORRection:Y:SPACing(?)

Determines whether the vertical, or amplitude, scaling is linear or logarithmic for interpolation of amplitude correction data.

Syntax

```
[:SENSe]:CORRection:Y:SPACing { LINear | LOGarithmic }
[:SENSe]:CORRection:Y:SPACing?
```

Arguments

`LINear` selects the linear scale for the interpolation.

`LOGarithmic` selects the logarithmic scale for the interpolation.

Measurement Modes

`SANORMAL`, `SASGRAM`

Examples

```
:SENSe:CORRection:Y:SPACing LINear
```
selects the linear scale for the interpolation.

[:SENSe]:EBWIDth Subgroup

The [:SENSe]:EBWidth commands set up the conditions related to the emission bandwidth (EBW) measurement.

Command Tree

Header	Parameter
[:SENSe]	
:EBWIDth	
:XDB	<numeric_value>

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   :INSTrument[:SELect] { SANORMAL | SASGRAM }
   ```

2. Run one of the following commands to start an EBW measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:EBWidth`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement EBWidth`

[:SENSe]:EBWidth:XDB(?)

Sets or queries the level relative to the maximum peak at which the EBW is measured (see Figure 2–19).

Syntax

```
[:SENSe]:EBWidth:XDB <rel_ampl>
[:SENSe]:EBWidth:XDB?
```

Arguments

<rel_ampl>::=<NRf> is the level at which the EBW is measured. Specify the amplitude relative to the maximum peak.
Range: –100 to –1 dB (default: –30 dB).

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:EBWidth:XDB -20
```

specifies that the EBW is measured at a level –20 dB lower than the maximum peak.

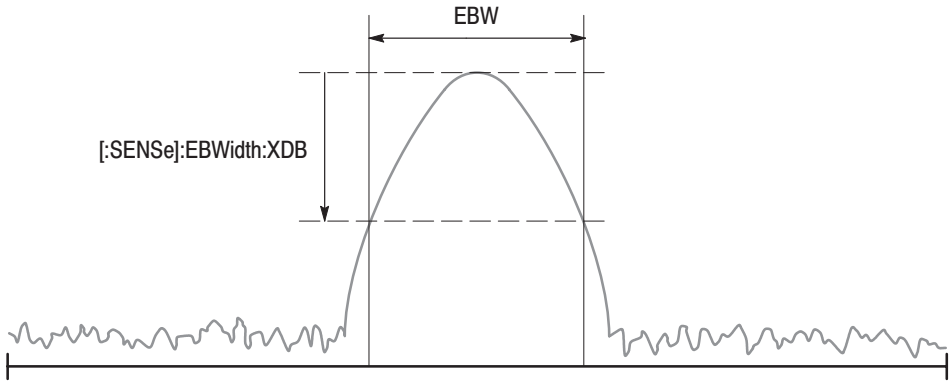


Figure 2–19: Setting up the EBW measurement

[:SENSe]:FEED Subgroup

The [:SENSe]:FEED commands select the input signal.

Command Tree

Header	Parameter
[:SENSe]	
:FEED	RF | AREFerence

[:SENSe]:FEED (No Query Form)

Selects the input signal: RF input or calibration signal.

Syntax

```
[:SENSe]:FEED { RF | AREFerence }
```

Arguments

RF selects the RF input.

AREFerence selects the internal calibration signal.

Measurement Modes

All

Examples

```
:SENSe:FEED RF
```

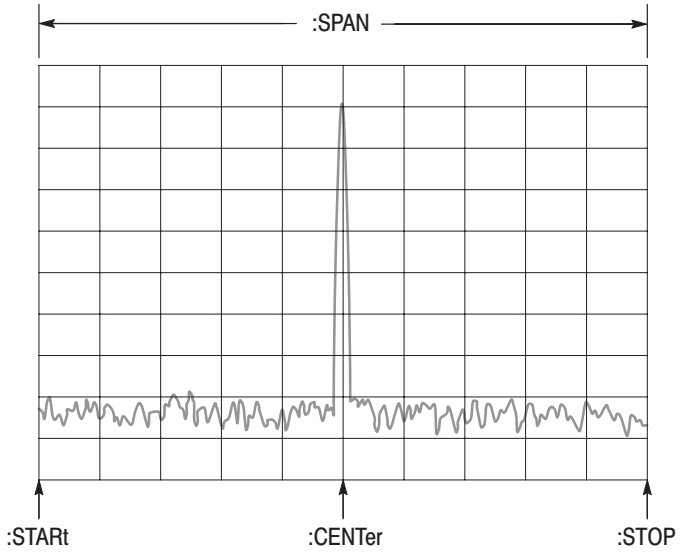
selects the RF input.

[:SENSe]:FREQuency Subgroup

The [:SENSe]:FREQuency commands set up the frequency-related conditions.

Command Tree

Header	Parameter
[:SENSe]	
:FREQuency	
:BAND?	
:CENTer	<frequency>
:STEP	<frequency>
:AUTO	<boolean>
[:INCRement]	<frequency>
:CHANnel	<numeric_value>
:CTABle	
:CATalog?	
[:SELect]	<table_name>
:SPAN	<frequency>
:STARt	<frequency>
:STOP	<frequency>



NOTE: Command header [:SENSe]:FREEQuency is omitted here.

Figure 2–20: Setting frequency and span

[:SENSe]:FREQuency:BAND? (Query Only)

Queries the measurement frequency band.

Syntax `[:SENSe]:FREQuency:BAND?`

Returns Table 2–39 shows the returned values and corresponding ranges:

Table 2–39: Measurement frequency bands

Argument	Frequency range
RF1B	10 MHz to 3 GHz (RSA2203A) 10 MHz to 3.5 GHz (RSA2208A)
RF2B	3.5 to 6.5 GHz (RSA2208A)
RF3B	5 to 8 GHz (RSA2208A)
BAS	DC to 20 MHz (Option 05)

Measurement Modes All

Examples `:SENSe:FREQuency:BAND?`
might return `RF1B`.

[:SENSe]:FREQuency:CENTer(?)

Sets or queries the center frequency.

Syntax

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

Arguments

<freq>::=<NRf> specifies the center frequency. For the setting range, refer to Table 2–39 on page 2–271.

Measurement Modes

All

Examples

```
:SENSe:FREQuency:CENTer 800MHz
```

sets the center frequency to 800 MHz.

Related Commands

[:SENSe]:FREQuency:BAND

[:SENSe]:FREQuency:CENTer:STEP:AUTO(?)

Determines whether to automatically set the step size (amount per click by which the up and down keys change a setting value) of the center frequency by the span setting.

Syntax

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

```
[:SENSe]:FREQuency:CENTer:STEP:AUTO?
```

Arguments

OFF or 0 specifies that the step size of the center frequency is not set automatically. To set it, use the [:SENSe]:FREQuency:CENTer:STEP[:INCRement] command.

ON or 1 specifies that the step size of the center frequency is set automatically by the span.

Measurement Modes

All

Examples

```
:SENSe:FREQuency:CENTer:STEP:AUTO ON
```

specifies that the step size of the center frequency is set automatically.

Related Commands

[:SENSe]:FREQuency:CENTer:STEP[:INCRement]

[:SENSe]:FREQuency:CENTer:STEP[:INCRement](?)

Sets or queries the step size (amount per click by which the up and down keys change a setting value) of the center frequency when [:SENSe]:FREQuency :CENTer:STEP:AUTO is OFF.

***NOTE**. This command is effective only in remote operation. It does not affect the front panel setting of the frequency step size.*

Syntax

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

Arguments

`<freq>::=<NRf>` is the step size of the center frequency.

Measurement Modes

All

Examples

```
:SENSe:FREQuency:CENTer:STEP:INCRement 10kHz
```

sets the step size of the center frequency to 10 kHz.

Related Commands

`[:SENSe]:FREQuency:CENTer:STEP:AUTO`

[:SENSe]:FREQuency:CHANnel(?)

Sets or queries a channel number in the channel table specified with the [:SENSe]:FREQuency:CTABle[:SELect] command.

Syntax

```
[:SENSe]:FREQuency:CHANnel <value>
[:SENSe]:FREQuency:CHANnel?
```

Arguments

<value>::=<NR1> specifies a channel number in the channel table.

Measurement Modes

All

Examples

```
:SENSe:FREQuency:CHANnel 10558
```

sets the channel number to 10558 for the W-CDMA downlink analysis.

Related Commands

[:SENSe]:FREQuency:CTABle[:SELect]

[:SENSe]:FREQuency:CTABle:CATalog? (Query Only)

Queries the available channel tables.

Syntax

```
[:SENSe]:FREQuency:CTABle:CATalog?
```

Returns

<string> is the available channel table name(s). If more than one table is available, the table names are separated with comma. Refer to the [:SENSe]:FREQuency:CTABle[:Select] command below for the table names.

Measurement Modes

All

Examples

```
:SENSe:FREQuency:CTABle:CATalog?
```

a partial return string may look like this:

```
"CDMA2000 EU PAMR400-FL","CDMA2000 EU PAMR400-RL","CDMA2000 EU
PAMR800-FL","CDMA2000 EU PAMR800-RL",...
```

Related Commands

[:SENSe]:FREQuency:CTABle[:SELect]

[:SENSe]:FREQuency:CTABle[:SELect](?)

Selects the channel table. The query command returns the selected channel table.

Syntax

```
[:SENSe]:FREQuency:CTABle[:SELect] <table>
[:SENSe]:FREQuency:CTABle[:SELect]?
```

Arguments

`<table>::=<string>` specifies a channel table. The table name is represented with the communication standard name followed by "-FL" (forward link), "-RL" (reverse link), "-UL" (uplink), or "-DL" (downlink).

The following channel tables are available:

None (does not use channel tables)

CDMA2000 EU PAMR400-FL	CDMA2000 EU PAMR400-RL
CDMA2000 EU PAMR800-FL	CDMA2000 EU PAMR800-RL
CDMA2000 GSM BAND 1-FL	CDMA2000 GSM BAND 1-RL
CDMA2000 GSM BAND 2-FL	CDMA2000 GSM BAND 2-RL
CDMA2000 IMT2000-FL	CDMA2000 IMT2000-RL
CDMA2000 JTACS BAND-FL	CDMA2000 JTACS BAND-RL
CDMA2000 KOREA PCS-FL	CDMA2000 KOREA PCS-RL
CDMA2000 N.A. 700MHz Cellular-FL	
CDMA2000 N.A. 700MHz Cellular-RL	
CDMA2000 N.A. Cellular-FL	CDMA2000 N.A. Cellular-RL
CDMA2000 N.A. PCS-FL	CDMA2000 N.A. PCS-RL
CDMA2000 NMT450 20k-FL	CDMA2000 NMT450 20k-RL
CDMA2000 NMT450 25k-FL	CDMA2000 NMT450 25k-RL
CDMA2000 SMR800-FL	CDMA2000 SMR800-RL
CDMA2000 TACS BAND-FL	CDMA2000 TACS BAND-RL
DCS1800-DL	DCS1800-UL
GSM850-DL	GSM850-UL
GSM900-DL	GSM900-UL
NMT450-DL	NMT450-UL
PCS1900-DL	PCS1900-UL
W-CDMA-DL	W-CDMA-UL

The table name must be within quotation marks for the argument.

Measurement Modes

All

Examples

```
:SENSe:FREQuency:CTABle:SELect "W-CDMA-DL"
```

selects the W-CDMA downlink channel table.

Related Commands

`[:SENSe]:FREQuency:CTABle:CATalog?`

[:SENSe]:FREQuency:SPAN(?)

Sets or queries the span.

***NOTE**. There are the following relationships among the center, start, and stop frequencies and the span; they are set interlinked manner:*

(Stop frequency + Start frequency) / 2 = Center frequency

Stop frequency – Start frequency = Span

When you set one of these, all the other settings are automatically changed correspondingly.

Syntax

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

Arguments

`<freq>::=<NRf>` specifies the span. The valid range depends on the measurement mode as listed in Table 2–40:

Table 2–40: Span setting

Measurement mode	Frequency band	Setting range
SANORMAL SASGRAM	RF	50 Hz to 3 GHz (continuous)
	Baseband (Option 05)	50 Hz to 20 MHz (continuous)
Other than above	RF	100 Hz to 10 MHz (1-2-5 sequence)
	Baseband (Option 05)	100 Hz to 20 MHz (1-2-5 sequence)

Measurement Modes

All

Examples

```
:SENSe:FREQuency:SPAN 1MHz
```

sets the span to 1 MHz.

Related Commands

`[:SENSe]:FREQuency:CENTer`, `[:SENSe]:FREQuency:STARt`, `[:SENSe]:FREQuency:STOP`

[:SENSe]:FREQuency:STARt(?)

Sets or queries the start frequency.

Syntax

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

Arguments

<freq>::=<NRf> specifies the start frequency. For the setting range, refer to Table 2–39 on page 2–271.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:FREQuency:STARt 800MHz
```

sets the start frequency to 800 MHz.

Related Commands

[:SENSe]:FREQuency:CENTer, [:SENSe]:FREQuency:SPAN, [:SENSe]:FREQuency:STOP

[:SENSe]:FREQuency:STOP(?)

Syntax

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

Arguments

<freq>::=<NRf> specifies the stop frequency. For the setting range, refer to Table 2–39 on page 2–271.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:FREQuency:STOP 1GHz
```

sets the stop frequency to 1 GHz.

Related Commands

[:SENSe]:FREQuency:CENTer, [:SENSe]:FREQuency:SPAN, [:SENSe]:FREQuency:STARt

[:SENSe]:OBWidth Subgroup

The [:SENSe]:OBWidth commands set the conditions related to the occupied bandwidth (OBW) measurement.

Command Tree

Header	Parameter
[SENSe]	
:OBWidth	
:PERCent	<numeric_value>

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   :INSTrument[:SELect] { SANORMAL | SASGRAM | SARTIME }
   ```

2. Run one of the following commands to start the OBW measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:OBWidth`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement OBWidth`

[:SENSe]:OBWidth:PERCent(?)

Sets or queries the occupied bandwidth for the OBW measurement.

Syntax

```
[:SENSe]:OBWidth:PERCent <value>
[:SENSe]:OBWidth:PERCent?
```

Arguments

<value>::=<NRf> specifies the occupied bandwidth.
Range: 80 to 99.99% (default: 99%)

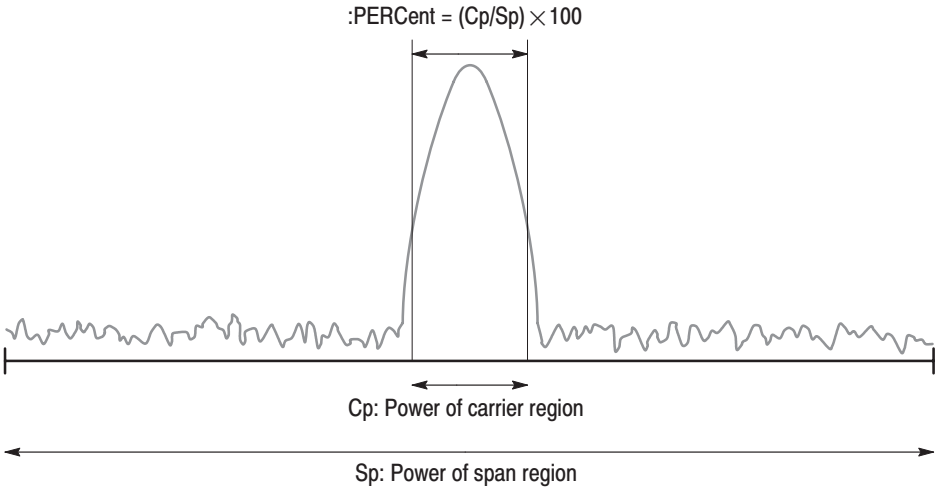
Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:OBWidth:PERCent 95
```

sets the occupied bandwidth to 95%.



NOTE: The command header [:SENSe]:OBWidth is omited here.

Figure 2–21: Setting up the OBW measurement

[:SENSe]:PULSe Subgroup

The [:SENSe]:PULSe commands set up the conditions related to the pulse charcteristics analysis.

***NOTE**. To use a command from this group, you must have selected TIMPULSE (pulse characteristics analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
[:SENSe]	
:PULSe	
:BLOCk	
:CHPower	
:BANDwidth|BWIDth	
:INTegration	<numeric_value>
:CRESolution	<numeric_value>
:EBWidth	
:XDB	<numeric_value>
:FFT	
:COEFficient	<numeric_value>
:WINDow	
[:TYPE]	NYQuist | BH4B
:FILTer	
:BANDwidth|BWIDth	<numeric_value>
:COEFficient	<numeric_value>
:MEASurement	OFF | GAUSsian
:FREQuency	
:OFFSet	<numeric_value>
:RECovery	FIRSt | USER | OFF
[:IMMediate]	
:OBWidth	
:PERCent	<numeric_value>
:PTOFfset	<numeric_value>
:THReshold	<numeric_value>

[:SENSe]:PULSe:BLOCk(?)

Sets or queries the number of the block to measure in the pulse characteristics analysis.

Syntax

```
[:SENSe]:PULSe:BLOCk <value>
[:SENSe]:PULSe:BLOCk?
```

Arguments

<value>::=<NR1> specifies the block number. Zero represents the latest block. Range: –M to 0 (M: the number of acquired blocks)

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:BLOCk -5
```

sets the block number to –5.

[:SENSe]:PULSe:CHPower:BANDwidth|:BWIDth:INTegration(?)

Sets or queries the channel bandwidth for the channel power measurement in the pulse characteristics analysis.

Syntax

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

Arguments

<value>::=<NRf> is the channel bandwidth for the channel power measurement. Range: (Bin bandwidth) × 8 to full span [Hz].
Refer to the *RSA2203A and RSA2208A User Manual* for the bin bandwidth.

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:CHPower:BANDwidth:INTegration 1.5MHz
```

sets the channel bandwidth to 1.5 MHz.

[:SENSe]:PULSe:CRESolution(?)

Sets or queries the frequency measurement resolution in the pulse characteristics analysis.

Syntax

```
[:SENSe]:PULSe:CRESolution <value>
[:SENSe]:PULSe:CRESolution?
```

Arguments

<value>::={ 1 | 10 | 100 | 1k | 10k | 100k | 1M } [Hz] specifies the frequency measurement resolution.

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:CRESolution 1kHz
```

sets the frequency measurement resolution to 1 kHz.

[:SENSe]:PULSe:EBWidth:XDB(?)

Sets or queries the level relative to the maximum peak at which the EBW is measured in the pulse characteristics analysis. Refer to the [:SENSe]:EBWidth :XDB command on page 2–268.

Syntax

```
[:SENSe]:PULSe:EBWidth:XDB <value>
[:SENSe]:PULSe:EBWidth:XDB?
```

Arguments

<value>::=<NRf> is the level at which the EBW is measured.
Specify the amplitude relative to the maximum peak.
Range: –100 to –1 dB (default: –30 dB)

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:EBWidth:XDB -20
```

specifies that the EBW is measured at a level –20 dB lower than the maximum peak.

Related Commands

[:SENSe]:EBWidth:XDB

[:SENSe]:PULSe:FFT:COEFficient(?)

Sets or queries the roll-off ratio when the FFT window type is Nyquist in the pulse characteristics analysis.

Syntax

`[:SENSe]:PULSe:FFT:COEFficient <value>`

`[:SENSe]:PULSe:FFT:COEFficient?`

Arguments

`<value>::=<NRf>` specifies the roll-off ratio. Range: 0.0001 to 1.0 (default: 0.2)

Measurement Modes

`TIMPULSE`

Examples

`:SENSe:PULSe:FFT:COEFficient 0.5`
sets the roll-off ratio to 0.5.

Related Commands

`[:SENSe]:PULSe:FFT:WINDow[:TYPE]`

[:SENSe]:PULSe:FFT:WINDow[:TYPE](?)

Selects or queries the FFT window type in the pulse characteristics analysis.

Syntax

`[:SENSe]:PULSe:FFT:WINDow[:TYPE] { NYQuist | BH4B }`

`[:SENSe]:PULSe:FFT:WINDow[:TYPE]?`

Arguments

`NYQuist` selects the Nyquist window.

`BH4B` selects the Blackman-Harris 4B type window.

Measurement Modes

`TIMPULSE`

Examples

`:SENSe:PULSe:FFT:WINDow:TYPE NYQuist`
selects the Nyquist window.

[:SENSe]:PULSe:FILTer:BANDwidth|BWIDth(?)

Sets or queries the bandwidth of the time measurement filter in the pulse characteristics analysis.

Syntax

```
[:SENSe]:PULSe:FILTer:BANDwidth|BWIDth <value>
[:SENSe]:PULSe:FILTer:BANDwidth|BWIDth?
```

Arguments

<value>::=<NRf> specifies the bandwidth of the time measurement filter. Range: Span/10 to full span.

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:FILTer:BANDwidth 1MHz
```

sets the bandwidth of the time measurement filter to 1 MHz.

[:SENSe]:PULSe:FILTer:COEFficient(?)

Sets or queries the α/BT value for the measurement filter when [:SENSe]:PULSe:FILTer:MEASurement is set to GAUSsian.

Syntax

```
[:SENSe]:PULSe:FILTer:COEFficient <value>
[:SENSe]:PULSe:FILTer:COEFficient?
```

Arguments

<value>::=<NRf> sets the α/BT value for the Gaussian measurement filter. Range: 0.0001 to 1 (default: 0.35)

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:FILTer:COEFficient 0.5
```

sets the α/BT value to 0.5.

Related Commands

[:SENSe]:PULSe:FILTer:MEASurement

[:SENSe]:PULSe:FILTer:MEASurement(?)

Selects or queries the measurement filter for the time measurement in the pulse characteristics analysis.

Syntax

```
[:SENSe]:PULSe:FILTer:MEASurement { OFF | GAUSsian }
[:SENSe]:PULSe:FILTer:MEASuerment?
```

Arguments

`OFF` specifies that no measurement filter is used.

`GAUSsian` selects the Gaussian filter.

Measurement Modes

`TIMPULSE`

Examples

```
:SENSe:PULSe:FILTer:MEASurement GAUSsian
```
selects the Gaussian filter.

[:SENSe]:PULSe:FREQuency:OFFSet(?)

Sets or queries the frequency offset for the pulse-pulse phase and the frequency deviation measurements in the pulse characteristics analysis.

This command is valid when [:SENSe]:PULSe:FREQuency:RECovery is set to USER. This query command is valid when [:SENSe]:PULSe:FREQuency:RECovery is set to FIRSt or USER.

Syntax

```
[:SENSe]:PULSe:FREQuency:OFFSet <value>
[:SENSe]:PULSe:FREQuency:OFFSet?
```

Arguments

`<value>::=<NRf>` specifies the frequency offset. Range: –10 to +10 MHz

Measurement Modes

`TIMPULSE`

Examples

```
:SENSe:PULSe:FREQuency:OFFSet 5MHz
```
sets the frequency offset to 5 MHz.

Related Commands

`[:SENSe]:PULSe:FREQuency:RECovery`

[:SENSe]:PULSe:FREQuency:RECovery(?)

Selects or queries the frequency recovery for the pulse-pulse phase and the frequency deviation measurements in the pulse characteristics analysis.

Syntax

```
[:SENSe]:PULSe:FREQuency:RECovery { FIRSt | USER | OFF }
```

```
[:SENSe]:PULSe:FREQuency:RECovery?
```

Arguments

FIRSt specifies that frequency correction is performed for all pulses based on the frequency error value calculated from the first pulse included in the analysis range. The calculated frequency error is shown in the Frequency Offset side key.

USER specifies that all pulses are corrected by the value set up by the [:SENSe]:PULSe:FREQuency:OFFSet command.

OFF disables frequency correction.

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:FREQuency:RECovery FIRSt
```

specifies that frequency correction is performed using the first pulse.

Related Commands

[:SENSe]:PULSe:FREQuency:OFFSet

[:SENSe]:PULSe[:IMMediate] (No Query Form)

Runs calculation for acquired data in the pulse characteristics analysis. To acquire data, use the :INITiate command.

Syntax `[:SENSe]:PULSe[:IMMediate]`

Arguments None

Measurement Modes `TIMPULSE`

Examples `:SENSe:PULSe:IMMediate`
runs calculation for acquired data.

Related Commands `:INITiate`

[:SENSe]:PULSe:OBWidth:PERcent(?)

Sets or queries OBW (Occupied Bandwidth) for the OBW measurement in the pulse characteristics analysis.

Syntax `[:SENSe]:PULSe:OBWidth:PERcent <value>`

`[:SENSe]:PULSe:OBWidth:PERcent?`

Arguments `<value>::=<NRf>` specifies the occupied bandwidth.
Range: 80 to 99.9% (default: 99%).

Measurement Modes `TIMPULSE`

Examples `:SENSe:PULSe:OBWidth:PERCent 95`
sets the occupied bandwidth to 95%.

[:SENSe]:PULSe:PTOFfset(?)

Sets or queries the time offset for the pulse-pulse phase measurement point.

Syntax

```
[:SENSe]:PULSe:PTOFfset <value>
[:SENSe]:PULSe:PTOFfset?
```

Arguments

<value>::=<NRf> specifies the time offset. Range: 0 to 1 s (the default is 0)
The default value is 0 (zero), that is, the measurement point is at the beginning of the pulse-on time.

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:PTOFfset 1.5m
```
Sets the time offset to 1.5 ms.

[:SENSe]:PULSe:THReshold(?)

Sets or queries the threshold level to detect pulses in acquired data.

Syntax

```
[:SENSe]:PULSe:THReshold <value>
[:SENSe]:PULSe:THReshold?
```

Arguments

<value>::=<NRf> specifies the threshold level.
Range: –100 to 0 dBc (the default is –3 dBc)

Measurement Modes

TIMPULSE

Examples

```
:SENSe:PULSe:THReshold –20
```
sets the threshold level to –20 dBc.

[:SENSe]:ROSCillator Subgroup

The [:SENSe]:ROSCillator commands set up the reference oscillator.

Command Tree

Header	Parameter
[:SENSe]	
:ROSCillator	
:SOURce	INTernal | EXTernal

[:SENSe]:ROSCillator:SOURce(?)

Selects or queries the reference oscillator.

Syntax

```
[:SENSe]:ROSCillator:SOURce { INTernal | EXTernal }
```

```
[:SENSe]:ROSCillator:SOURce?
```

Arguments

INTernal selects the internal reference oscillator.

EXTernal selects the external reference oscillator. Connect it to the REF IN connector on the rear panel.

Measurement Modes

All

Examples

```
:SENSe:ROSCillator:SOURce EXTernal
```

selects the external reference oscillator.

[:SENSe]:SPECtrum Subgroup

The [:SENSe]:SPECtrum commands set up the conditions related to the spectrum measurement in the S/A (spectrum analysis) mode.

Command Tree

Header	Parameter
[:SENSe]	
:SPECtrum	
:AVERage	
:CLEar	
:COUNt	<numeric_value>
[:STATE]	<boolean>
TYPE	RMS | MAXimum | MINimum
:BANDwidth|:BWIDth	
[:RESolution]	<numeric_value>
:AUTO	<boolean>
:STATe	<boolean>
:DETector	
[:FUNction]	NEGative | POSitive | PNEgative
:FILTer	
:COEFficient	<numeric_value>
:TYPE	RECTangle | GAUSsian | NYQuist | RNYQuist
:FFT	
:ERESolution	<boolean>
:LENGth	<numeric_value>
:STARt	<numeric_value>
:WINDow	
[:TYPE]	BH3A | BH3B | BH4A | BH4B | BLACkman | HAMMing | HANNing | PARZen | ROSEnfield | WELCh | SLOBe | SCUBed | STO4 | FLATtop | RECT
:FRAMe	<numeric_value>
:MEASurement	OFF | CHPower | ACPower | OBWidth | EBWidth | CNRatio | CFRequency
:ZOOM	
:BLOCk	<numeric_value>
:FREQuency	
:CENTer	<numeric_value>
:WIDTh	<numeric_value>
:LENGth	<numeric_value>
:OFFSet	<numeric_value>

[:SENSe]:SPECtrum:AVERage:CLEar (No Query Form)

Clears average data and counter, and restarts the average process.

Syntax `[:SENSe]:SPECtrum:AVERage:CLEar`

Arguments None

Measurement Modes SANORMAL, SASGRAM

Examples `:SENSe:SPECtrum:AVERage:CLEar`
Clears average data and counter, and restarts the average process.

[:SENSe]:SPECtrum:AVERage:COUNt(?)

Sets or queries the number of traces to combine using the :TYPE setting (refer to page 2–293).

Syntax `[:SENSe]:SPECtrum:AVERage:COUNt <value>`

`[:SENSe]:SPECtrum:AVERage:COUNt?`

Arguments `<value>::=<NR1>` is the number of traces to combine for averaging.
Range: 1 to 10000 (default: 20)

Measurement Modes SANORMAL, SASGRAM

Examples `:SENSe:SPECtrum:AVERage:COUNt 64`
sets the average count to 64.

Related Commands `[:SENSe]:SPECtrum:AVERage:TYPE`

[:SENSe]:SPECtrum:AVERage[:STATe](?)

Determines whether to turn averaging on or off.

Syntax

```
[:SENSe]:SPECtrum:AVERage[:STATe] { OFF | ON | 0 | 1 }
[:SENSe]:SPECtrum:AVERage[:STATe]?
```

Arguments

OFF or 0 turns off averaging.

ON or 1 turns on averaging.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:AVERage:STATe ON
```

turns on averaging.

[:SENSe]:SPECtrum:AVERage:TYPE(?)

Selects or queries the type of averaging.

Syntax

```
[:SENSe]:SPECtrum:AVERage:TYPE { RMS | MAXimum | MINimum }
[:SENSe]:SPECtrum:AVERage:TYPE?
```

Arguments

RMS performs the average process with RMS (root-mean-square).

MAXimum retains the maximum value at each data point on the waveform.

MINimum retains the minimum value at each data point on the waveform.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:AVERage:TYPE RMS
```

performs the average process with RMS.

[:SENSe]:SPECtrum:BANDwidth|:BWIDth[:RESolution](?)

Sets or queries the resolution bandwidth (RBW) when [:SENSe]:SPECtrum:BANDwidth|:BWIDth[:RESolution]:AUTO is set to Off.

Syntax

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

Arguments

<freq>::=<NRf> specifies the RBW.
For the setting range, refer to Table D–2 in *Appendix D*.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:BANDwidth:RESolution 80kHz
```
sets the RBW to 80 kHz.

[:SENSe]:SPECtrum:BANDwidth|:BWIDth[:RESolution]:AUTO(?)

Determines whether to automatically set the resolution bandwidth (RBW) by the span setting.

Syntax

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

Arguments

OFF or 0 specifies that the RBW is not set automatically. To set it, use the [:SENSe]:SPECtrum:BANDwidth|:BWIDth[:RESolution] command.

ON or 1 specifies that the RBW is set automatically.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:BANDwidth:RESolution:AUTO ON
```
specifies that the RBW is set automatically.

Related Commands

:INSTrument[:SELect]

[:SENSe]:SPECtrum:BANDwidth|:BWIDth:STATe(?)

Determines whether to perform the resolution bandwidth (RBW) process.

Syntax

```
[:SENSe]:SPECtrum:BANDwidth|:BWIDth:STATe { OFF | ON | 0 | 1 }
[:SENSe]:SPECtrum:BANDwidth|:BWIDth:STATe?
```

Arguments

OFF or 0 specifies that the RBW process is not performed so that a spectrum immediately after the FFT process is displayed on screen.

ON or 1 specifies that the RBW process is performed.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:BANDwidth:STATe ON
```

specifies that the resolution bandwidth process is performed.

[:SENSe]:SPECtrum:DETector[:FUNCtion](?)

Selects or queries the display detector (method to be used for decimating traces to fit the available horizontal space on screen).

The number of horizontal pixel positions on screen is generally smaller than that of waveform data points. When actually displayed, the waveform data is therefore thinned out according to the number of horizontal pixel positions which can be displayed. For the details, refer to the *RSA2203A and RSA2208A User Manual*.

Syntax

```
[:SENSe]:SPECtrum:DETector[:FUNCtion] { NEGative | POSitive
| PNEgative }
```

```
[:SENSe]:SPECtrum:DETector[:FUNCtion]?
```

Arguments

`NEGative` shows the minimum value of the data corresponding to each horizontal pixel position.

`POSitive` shows the maximum value of the data corresponding to each horizontal pixel position.

`PNEgative` draws a line connecting the maximum and minimum points of the data corresponding to each horizontal pixel position.

Measurement Modes

`SANORMAL`, `SASGRAM`

Examples

```
:SENSe:SPECtrum:DETector:FUNCtion PNEgative
```

displays waveform drawing a line that connects the maximum and minimum points of the data for each pixel.

[:SENSe]:SPECtrum:FILTer:COEFficient(?)

Sets or queries the roll-off rate of the RBW filter when you have selected either NYQuist (Nyquist filter) or RNYQuist (Root Nyquist filter) in the [:SENSe]:SPECtrum:FILTer:TYPE command.

Syntax

```
[:SENSe]:SPECtrum:FILTer:COEFficient <ratio>
[:SENSe]:SPECtrum:FILTer:COEFficient?
```

Arguments

`<ratio>::=<NRf>` specifies the roll-off rate. Range: 0 to 1.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:FILTer:COEFficient 0.5
```
sets the RBW filter roll-off rate to 0.5.

Related Commands

`[:SENSe]:SPECtrum:FILTer:TYPE`

[:SENSe]:SPECtrum:FILTer:TYPE(?)

Selects or queries the RBW filter.

Syntax

```
[:SENSe]:SPECtrum:FILTer:TYPE { RECTangle | GAUSsian | NYQuist
| RNYQuist }
[:SENSe]:SPECtrum:FILTer:TYPE?
```

Arguments

`RECTangle` selects the rectangular filter.

`GAUSsian` selects the Gaussian filter.

`NYQuist` selects the Nyquist filter (default).

`RNYQuist` selects the Root Nyquist filter.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:FILTer:TYPE NYQuist
```
selects the Nyquist filter for RBW.

[:SENSe]:SPECtrum:FFT:ERESolution(?)

Determines whether to enable the extended resolution that eliminates the limit on the number of FFT points (it is normally limited internally).

Syntax

```
[:SENSe]:SPECtrum:FFT:ERESolution { OFF | ON | 0 | 1 }
[:SENSe]:SPECtrum:FFT:ERESolution?
```

Arguments

OFF or 0 disables the extended resolution. The number of FFT points is limited internally.

ON or 1 allows you to set the number of FFT points up to 65536. Use the [:SENSe]:SPECtrum:FFT:LENGth command to set the number.

***NOTE**. It is recommended to keep the extended resolution off as its default condition.*

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:FFT:ERESolution ON
```

enables the extended resolution.

Related Commands

[:SENSe]:SPECtrum:FFT:LENGth

[:SENSe]:SPECtrum:FFT:LENGth(?)

Sets or queries the number of FFT points. This command is valid when [:SENSe]:SPECtrum:BANDwidth|:BWIDth:STATe is OFF.

Syntax

```
[:SENSe]:SPECtrum:FFT:LENGth <value>
[:SENSe]:SPECtrum:FFT:LENGth?
```

Arguments

<value>::=<NR1> sets the number of FFT points.
Range: 64 to 65536 in powers of 2.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:FFT:LENGth 1024
```
sets the number of FFT points to 1024.

Related Commands

[:SENSe]:SPECtrum:BWIDth:STATe, [:SENSe]:SPECtrum:FFT:ERESolution

[:SENSe]:SPECtrum:FFT:STARt(?)

Sets or queries the start point of the 1024-point FFT frame by the number of samples from the previous frame for the FFT overlap.

***NOTE**. This command is valid when :INSTrument[:SELect] is set to SARTIME (Real Time S/A).*

Syntax

```
[:SENSe]:SPECtrum:FFT:STARt <value>
[:SENSe]:SPECtrum:FFT:STARt?
```

Arguments

<value>::={ 64 | 128 | 256 | 512 | 1024 } selects the start point of the 1024-point FFT frame by the number of samples from the previous frame.

Measurement Modes

SARTIME

Examples

```
:SENSe:SPECtrum:FFT:STARt 256
```
sets the start point of the 1024-point FFT frame to 256 samples.

[:SENSe]:SPECtrum:FFT:WINDow[:TYPE](?)

Selects or queries the FFT window function.

Syntax

```
[:SENSe]:SPECtrum:FFT:WINDow[:TYPE] { BH3A | BH3B | BH4A | BH4B
| BLACkman | HAMMing | HANNing | PARZen | ROSenfield | WELCh
| SLOBe | SCUBed | ST4T | FLATtop | RECT }

[:SENSe]:SPECtrum:FFT:WINDow[:TYPE]?
```

Arguments

Table 2–41 shows the arguments and their meanings.

Table 2–41: FFT windows

Argument	FFT window
BH3A	Blackman–Harris 3A type
BH3B	Blackman–Harris 3B type
BH4A	Blackman–Harris 4A type
BH4B	Blackman–Harris 4B type
BLACkman	Blackman
HAMMing	Hamming
HANNing	Hanning
PARZen	Parzen
ROSenfield	Rosenfield
WELCh	Welch
SLOBe	Sine lobe
SCUBed	Sine cubed
ST4T	Sine to 4th
FLATtop	Flat top
RECT	Rectangular

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPECtrum:FFT:WINDow:TYPE HAMMing
```

selects the Hamming window.

[:SENSe]:SPECtrum:FRAMe(?)

Sets or queries the frame number of the spectrum frame to be measured in the Real Time S/A (real-time spectrum analysis) mode.

Syntax

```
[:SENSe]:SPECtrum:FRAMe <number>
[:SENSe]:SPECtrum:FRAMe?
```

Arguments

<number>::=<NR1> specifies the frame number. Range: –M to 0
(M: Block size set with the [:SENSe]:BSIZe command)

Measurement Modes

SARTIME

Examples

```
:SENSe:SPECtrum:FRAMe -5
```
sets the frame number to –5.

Related Commands

[:SENSe]:BSIZe, [:SENSe]:SPECtrum:BLOCk

[:SENSe]:SPECtrum:MEASurement(?)

Selects and runs the measurement item in the S/A (spectrum analysis) mode. The query version of this command returns the current measurement item.

Syntax

```
[:SENSe]:SPECtrum:MEASurement { OFF | CHPower | ACPower | OBWidth
| EBWidth | CNRatio | CFRequency | SPURious }

[:SENSe]:SPECtrum:MEASurement?
```

Arguments

Table 2–42 shows the arguments and their meanings.

Table 2–42: S/A mode measurement items

Argument	Measurement item
OFF	Turns off the measurement.
CHPower	Channel power
ACPower	Adjacent channel leakage power (ACPR)
OBWidth	Occupied bandwidth (OBW)
EBWidth	Emission bandwidth (EBW)
CNRatio	Carrier-to-noise ratio (C/N)
CFRequency	Carrier frequency
SPURious	Spurious signal

Measurement Modes

SANORMAL, SASGRAM, SARTIME

Examples

```
:SENSe:SPECtrum:MEASurement CHPower
```

runs the channel power measurement.

[:SENSe]:SPECtrum:ZOOM:BLOCk(?)

Sets or queries the number of the block to zoom in the Real-Time S/A with Zoom mode.

Syntax

```
[:SENSe]:SPECtrum:ZOOM:BLOCk <value>
```

```
[:SENSe]:SPECtrum:ZOOM:BLOCk?
```

Arguments

<number>::=<NR1> specifies the block number to zoom.
Zero represents the latest block.
Range: –M to 0 (M: Number of acquired blocks).

Measurement Modes

SAZRTIME

Examples

```
:SENSe:SPECtrum:ZOOM:BLOCk -5
```

sets the block number to –5.

[:SENSe]:SPECtrum:ZOOM:FREQuency:CENTer(?)

Sets or queries the center frequency of a zoomed area in the Real-Time S/A with Zoom mode.

Syntax

```
[:SENSe]:SPECtrum:ZOOM:FREQuency:CENTer <value>
[:SENSe]:SPECtrum:ZOOM:FREQuency:CENTer?
```

Arguments

<value>::=<NRf> specifies the center frequency of a zoomed area. The setting value must be within the measurement frequency range.

Measurement Modes

SAZRTIME

Examples

```
:SENSe:SPECtrum:ZOOM:FREQuency:CENTer 1.75GHz
```

sets the center frequency of the zoomed area to 1.75 GHz.

[:SENSe]:SPECtrum:ZOOM:FREQuency:WIDTh(?)

Sets or queries the frequency width of a zoomed area in the Real-Time S/A with Zoom mode.

Syntax

```
[:SENSe]:SPECtrum:ZOOM:FREQuency:WIDTh <value>
[:SENSe]:SPECtrum:ZOOM:FREQuency:WIDTh?
```

Arguments

<value>::=<NRf> specifies the frequency width of a zoomed area. The setting value must be within the measurement frequency range.

Measurement Modes

SAZRTIME

Examples

```
:SENSe:SPECtrum:ZOOM:FREQuency:WIDTh 500kHz
```

sets the frequency width of the zoomed area to 500 kHz.

[:SENSe]:SPECtrum:ZOOM:LENGth(?)

Sets or queries the time length of a zoomed area in the Real-Time S/A with Zoom mode.

Syntax

```
[:SENSe]:SPECtrum:ZOOM:LENGth <value>
[:SENSe]:SPECtrum:ZOOM:LENGth?
```

Arguments

<value>::=<NR1> specifies the range of a zoomed area by the number of data points. Range: 1 to [1024 × (block size)] or [81920 – 512 = 81408] whichever smaller. To set the block size, use the [:SENSe]:BSIZe command.

Measurement Modes

SAZRTIME

Examples

```
:SENSe:SPECtrum:ZOOM:LENGth 1000
```

sets the measurement range to 1000 points.

Related Commands

[:SENSe]:BSIZe

[:SENSe]:SPECtrum:ZOOM:OFFSet(?)

Sets or queries the starting point of a zoomed area in the Real-Time S/A with Zoom mode.

Syntax

```
[:SENSe]:SPECtrum:ZOOM:OFFSet <value>
[:SENSe]:SPECtrum:ZOOM:OFFSet?
```

Arguments

<value>::=<NRf> specifies the starting point of a zoomed area by considering the trigger output point as the reference. Range: 0 to 1024 × (Block size) –1. To set the block size, use the [:SENSe]:BSIZe command.

Measurement Modes

SAZRTIME

Examples

```
:SENSe:SPECtrum:ZOOM:OFFSet 500
```

sets the starting point of a zoomed area to point 500.

Related Commands

[:SENSe]:BSIZe

[:SENSe]:SPURious Subgroup

The [:SENSe]:SPURious commands set up the conditions related to the spurious signal measurement.

Command Tree

Header	Parameter
[SENSe]	
:SPURious	
[:THReshold]	
:EXCursion	<numeric_value>
:IGNore	<numeric_value>
:SIGNal	<numeric_value>
:SPURious	<numeric_value>

Prerequisites for Use

To use a command of this group, you must have run at least the following two commands:

1. Run the following command to set the measurement mode to S/A:

   ```
   :INSTrument[:SELect] { SANORMAL | SASGRAM }
   ```

2. Run one of the following commands to start the spurious signal measurement:

   - To start the measurement with the default settings:
     `:CONFigure:SPECtrum:SPURious`
   - To start the measurement without modifying the current settings:
     `[:SENSe]:SPECtrum:MEASurement SPURious`

[:SENSe]:SPURious[:THReshold]:EXCursion(?)

Sets or queries the excursion level to determine if the signal is spurious in the spurious signal measurement (see Figure 2–22).

Syntax

```
[:SENSe]:SPURious[:THReshold]:EXCursion <level>
[:SENSe]:SPURious[:THReshold]:EXCursion?
```

Arguments

<level>::=<NRf> specifies the excursion level. If the signal exceeds the excursion level and meets the other threshold requirements that you set, it is decided to be spurious. Range: 0 to 30 dB (default: 3dB)

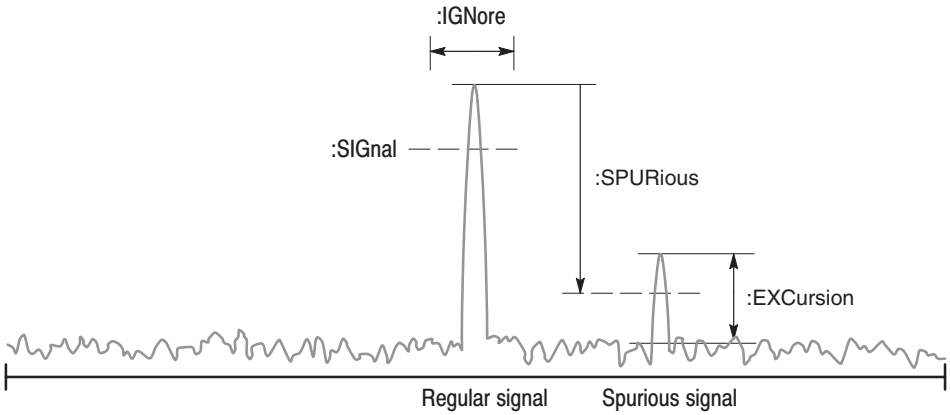
Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPURious:THReshold:EXCursion 5
```

sets the excursion level to 5 dB.



NOTE: Command header [:SENSe]:SPURious[:THReshold] is omitted here.

Figure 2–22: Setting up the spurious signal measurement

[:SENSe]:SPURious[:THReshold]:IGNore(?)

Sets or queries the range not to detect spurious signals around the carrier peak signal to avoid mistaking spurious (see Figure 2–22).

Syntax

```
[:SENSe]:SPURious[:THReshold]:IGNore <value>
[:SENSe]:SPURious[:THReshold]:IGNore?
```

Arguments

<value>::=<NRf> specifies the range not to detect spurious around the carrier peak signal. Range: 0 to Span/2 [Hz].

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPURious:THReshold:IGNore 1MHz
```

sets the range not to detect spurious to 1 MHz.

[:SENSe]:SPURious[:THReshold]:SIGNal(?)

Sets or queries the threshold level to determine if the signal is the carrier in the spurious signal measurement (see Figure 2–22).

Syntax

```
[:SENSe]:SPURious[:THReshold]:SIGNal <level>
[:SENSe]:SPURious[:THReshold]:SIGNal?
```

Arguments

<level>::=<NR1> specifies the signal criterion level. If the signal exceeds the level, it is decided to be the carrier. Range: –100 to +30 dBm

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPURious:THReshold:SIGNal -30
```

sets the carrier criterion level to –30 dBm.

[:SENSe]:SPURious[:THReshold]:SPURious(?)

Sets or queries the threshold level to determine if the signal is spurious in the spurious signal measurement (see Figure 2–22).

Syntax

```
[:SENSe]:SPURious[:THReshold]:SPURious <level>
```

```
[:SENSe]:SPURious[:THReshold]:SPURious?
```

Arguments

<level>::=<NR1> specifies the spurious criterion level relative to the carrier peak. If the signal exceeds the level and meets the other threshold requirements that you set, it is decided to be spurious. Range: –90 to –30 dB.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:SENSe:SPURious:THReshold:SPURious -50
```

sets the spurious criterion level to –50 dB relative to the carrier peak.

[:SENSe]:TRANsient Subgroup

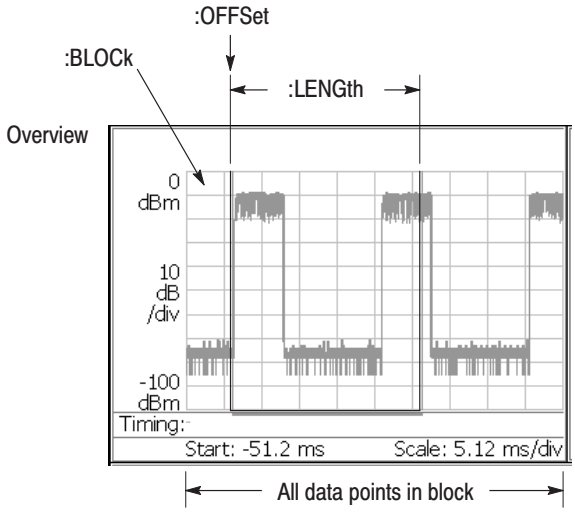
The [:SENSe]:TRANsient commands set up the conditions related to the time characteristic analysis. The time characteristic analysis includes IQ level vs. time, power vs. time, and frequency vs. time measurements.

***NOTE**. To use a command of this group, you must have selected TRAN (time characteristic analysis) in the :INSTrument[:SELect] command.*

Command Tree

Header	Parameter
[:SENSe]	
:TRANsient	
:BLOCk	<numeric_value>
[:IMMediate]	
:ITEM	IQVTime | PVTime | FVTime
:LENGth	<numeric_value>
:OFFSet	<numeric_value>

For the commands defining the analysis range, see the figure below. The analysis range is shown by a green line in the overview.



NOTE: Command header [:SENSe]:TRANsient is omitted here.

Figure 2–23: Defining the analysis range

[:SENSe]:TRANsient:BLOCk(?)

Sets or queries the number of the block to measure in the time characteristic analysis.

Syntax

```
[:SENSe]:TRANsient:BLOCk <value>
[:SENSe]:TRANsient:BLOCk?
```

Arguments

<value>::=<NR1> specifies the block number. Zero represents the latest block. Range: –M to 0 (M: Number of acquired blocks)

Measurement Modes

TIMTRAN

Examples

```
:SENSe:TRANsient:BLOCk -5
```

sets the block number to –5.

[:SENSe]:TRANsient[:IMMediate] (No Query Form)

Runs the time characteristic analysis calculation for the acquired data.
To select the measurement item, use the [:SENSe]:TRANsient:ITEM command.
To acquire data, use the :INITiate command.

Syntax

```
[:SENSe]:TRANsient[:IMMediate]
```

Arguments

None

Measurement Modes

TIMTRAN

Examples

```
:SENSe:TRANsient:IMMediate
```

runs the time characteristic analysis calculation.

Related Commands

:INITiate, [:SENSe]:TRANsient:ITEM

[:SENSe]:TRANsient:ITEM(?)

Selects or queries the measurement item in the time characteristic analysis.

Syntax

```
[:SENSe]:TRANsient:ITEM { OFF | IQVTime | PVTime | FVTime }
[:SENSe]:TRANsient:ITEM?
```

Arguments

OFF turns off measurement.

IQVTime selects the IQ level vs. time measurement.

PVTime selects the power vs. time measurement.

FVTime selects the frequency vs. time measurement.

Measurement Modes

TIMTRAN

Examples

```
:SENSe:TRANsient:ITEM IQVTime
```

selects the IQ level vs. time measurement.

[:SENSe]:TRANsient:LENGth(?)

Sets or queries the range for the time characteristic analysis.

Syntax

```
[:SENSe]:TRANsient:LENGth <value>
[:SENSe]:TRANsient:LENGth?
```

Arguments

<value>::=<NR1> specifies the analysis range by the number of data points. Range: 1 to 1024 × (Block size). To set the block size, use the [:SENSe]:BSIZe command.

Measurement Modes

TIMTRAN

Examples

```
:SENSe:TRANsient:LENGth 1000
```

sets the analysis range to 1000 points.

Related Commands

[:SENSe]:BSIZe

[:SENSe]:TRANsient:OFFSet(?)

Sets or queries the measurement start position in the time characteristic analysis.

Syntax

```
[:SENSe]:TRANsient:OFFSet <value>
[:SENSe]:TRANsient:OFFSet?
```

Arguments

<value>::=<NR1> defines the measurement start position by the number of points. Range: 0 to 1024 × (Block size). To set the block size, use the [:SENSe]:BSIZe command.

Measurement Modes

TIMTRAN

Examples

```
:SENSe:TRANsient:OFFSet 500
```

sets the measurement start position to point 500.

Related Commands

[:SENSe]:BSIZe

:STATus Commands

The :STATus commands control the SCPI-defined status reporting structures. In addition to those in IEEE 488.2, the analyzer has questionable and operation registers defined in SCPI. These registers conform to the IEEE 488.2 specification and each is comprised of a condition register, an event register, an enable register, and negative and positive transition filters. For details on these registers, refer to *Status and Events* beginning on page 3–1.

Command Tree

Header	Parameter
:STATus	
:OPERation	
:CONDition	
:ENABle	<bit_value>
[:EVENt]?	
:NTRansition	<bit_value>
:PTRansition	<bit_value>
:PRESet	
:QUEStionable	
:CONDition	
:ENABle	<bit_value>
[:EVENt]?	
:NTRansition	<bit_value>
:PTRansition	<bit_value>

:STATus:OPERation:CONDition? (Query Only)

Returns the contents of the Operation Condition Register (OCR). For detail on the register, refer to Chapter 3, *Status and Events*.

Syntax

```
:STATus:OPERation:CONDition?
```

Arguments

None

Returns

<NR1> is a decimal number showing the contents of the OCR.

Measurement Modes

All

Examples

```
:STATus:OPERation:CONDition?
```

might return 16, showing that the bits in the OCR have the binary value 0000000000010000, which means the analyzer is in measurement.

:STATus:OPERation:ENABle (?)

Sets or queries the enable mask of the Operation Enable Register (OENR) which allows true conditions in the Operation Event Register to be reported in the summary bit. For detail on the register, refer to Chapter 3, *Status and Events*.

Syntax

```
:STATus:OPERation:ENABle <bit_value>
```

```
:STATus:OPERation:ENABle?
```

Arguments

<bit_value>::=<NR1> is the enable mask of the OENR. Range: 0 to 65535.

Returns

<NR1> is a decimal number showing the contents of the OENR.
Range: 0 to 32767 (The most-significant bit cannot be set true.)

Measurement Modes

All

Examples

```
:STATus:OPERation:ENABle 1
```

enables the CALibrating bit.

```
:STATus:OPERation:ENABle?
```

might return 1, showing that the bits in the OENR have the binary value 00000000 00000001, which means that the CAL bit is valid.

:STATus:OPERation[:EVENt]? (Query Only)

Returns the contents of the Operation Event Register (OEVR). Reading the OEVR clears it. For detail on the register, refer to Chapter 3, *Status and Events*.

Syntax

```
:STATus:OPERation[:EVENt]?
```

Arguments

None

Returns

<NR1> is a decimal number showing the contents of the OEVR.

Measurement Modes

All

Examples

```
STATus:OPERation:EVENt?
```

might return 1, showing that the bits in the OEVR have the binary value 00000000 00000001, which means that the CAL bit is set.

:STATus:OPERation:NTRansition (?)

Sets or queries the negative transition filter value of the Operation Transition Register (OTR). For detail on the register, refer to Chapter 3, *Status and Events*.

Syntax

```
:STATus:OPERation:NTRansition <bit_value>
```

```
:STATus:OPERation:NTRansition?
```

Arguments

<bit_value>::=<NR1> is the negative transition filter value. Range: 0 to 65535.

Returns

<NR1> is a decimal number showing the contents of the OTR.
Range: 0 to 32767 (The most-significant bit cannot be set true.)

Measurement Modes

All

Examples

```
:STATus:OPERation:NTRansition #H120
```

sets the negative transition filter value to #H120.

```
:STATus:OPERation:NTRansition?
```

might return 288.

:STATus:OPERation:PTRansition (?)

Sets or queries the positive transition filter value of the Operation Transition Register (OTR). For detail on the register, refer to Chapter 3, *Status and Events*.

Syntax

```
:STATus:OPERation:PTRansition <bit_value>
```

```
:STATus:OPERation:PTRansition?
```

Arguments

<bit_value>::=<NR1> is the positive transition filter value. Range: 0 to 65535.

Returns

<NR1> is a decimal number showing the contents of the OTR.
Range: 0 to 32767 (The most-significant bit cannot be set true.)

Measurement Modes

All

Examples

```
:STATus:OPERation:PTRansition 0
```
sets the positive transition filter value to 0.

```
:STATus:OPERation:PTRansition?
```
might return 0.

:STATus:PRESet (No Query Form)

Presets SCPI enable registers OENR (Operation Enable Register) and QENR (Questionable Enable Register). For details on the registers, refer to Chapter 3, *Status and Events*.

Syntax

```
:STATus:PRESet
```

Arguments

None

Measurement Modes

All

Examples

```
:STATus:PRESet
```
presets the registers OENR and QENR.

:STATus:QUEStionable:CONDition? (Query Only)

Returns the contents of the Questionable Condition Register (QCR). For detail on the register, refer to Chapter 3, *Status and Events*.

***NOTE**. The QCR is not used in the RSA2203A/RSA2208A analyzer.*

Syntax `:STATus:QUEStionable:CONDition?`

Arguments None

Returns <NR1> is a decimal number showing the contents of the QCR.

Measurement Modes All

:STATus:QUEStionable:ENABle (?)

Sets or queries the enable mask of the Questionable Enable Register (QENR) which allows true conditions in the Questionable Event Register to be reported in the summary bit. For detail on the register, refer to Chapter 3, *Status and Events*.

***NOTE**. The QENR is not used in the RSA2203A/RSA2208A analyzer.*

Syntax `:STATus:QUEStionable:ENABle <bit_value>`

`:STATus:QUEStionable:ENABle?`

Arguments <bit_value>::=<NR1> is the enable mask of QENR. Range: 0 to 65535.

Returns <NR1> is a decimal number showing the contents of the QENR. Range: 0 to 32767 (The most-significant bit cannot be set true.)

Measurement Modes All

:STATus:QUEStionable[:EVENt]? (Query Only)

Returns the contents of the Questionable Event Register (QEVR). Reading the QEVR clears it. For detail on the register, refer to Chapter 3, *Status and Events*.

***NOTE**. The QEVR is not used in the RSA2203A/RSA2208A analyzer.*

Syntax

```
:STATus:QUEStionable[:EVENt]?
```

Arguments None

Returns <NR1> is a decimal number showing the contents of the QEVR.

Measurement Modes All

:STATus:QUEStionable:NTRansition (?)

Sets or queries the negative transition filter value of the Operation Transition Register (QTR). For detail on the register, refer to Chapter 3, *Status and Events*.

***NOTE**. The QTR is not used in the RSA2203A/RSA2208A analyzer.*

Syntax

```
:STATus:QUEStionable:NTRansition <bit_value>
:STATus:QUEStionable:NTRansition?
```

Arguments <bit_value>::=<NR1> is the negative transition filter value. Range: 0 to 65535.

Returns <NR1> is a decimal number showing the contents of the QTR.
Range: 0 to 32767 (The most-significant bit cannot be set true.)

Measurement Modes All

:STATus:QUEStionable:PTRansition (?)

Sets or queries the positive transition filter value of the Questionable Transition Register (QTR). For detail on the register, refer to Chapter 3, *Status and Events*.

***NOTE**. The QTR is not used in the RSA2203A/RSA2208A analyzer.*

Syntax

```
:STATus:QUEStionable:PTRansition <bit_value>
:STATus:QUEStionable:PTRansition?
```

Arguments

<bit_value>::=<NR1> is the positive transition filter value. Range: 0 to 65535.

Returns

<NR1> is a decimal number showing the contents of the QTR.
Range: 0 to 32767 (The most-significant bit cannot be set true.)

Measurement Modes

All

:SYSTem Commands

The :SYSTem commands set up the system-related conditions.

Command Tree

Header	Parameter
:SYSTem	
:DATE	<year>,<month>,<day>
:ERRor	
:ALL?	
:CODE	
:ALL?	
[:NEXT]?	
:COUNt?	
[:NEXT]?	
:KLOCk	<boolean>
:OPTions?	
:PRESet	
:TIME	<hour>,<minute>,<second>
:VERSion?	

:SYSTem:DATE (?)

Sets or queries the date (year, month, and day). This command is equivalent to the date setting through the Windows Control Panel.

Syntax

```
:SYSTem:DATE <year>,<month>,<day>
:SYSTem:DATE?
```

Arguments

<year>::=<NRf> specifies the year (4 digits). Range: 2000 to 2099

<month>::=<NRf> specifies the month. Range: 1 (January) to 12 (December)

<day>::=<NRf> specifies the day. Range: 1 to 31

These values are rounded to the nearest integer.
*RST has no effect on the settings.

Measurement Modes

All

Examples

```
:SYSTem:DATE 2002,3,19
```
sets the internal calendar to March 19, 2002.

Related Commands

:SYSTem:TIME

:SYSTem:ERRor:ALL? (Query Only)

Returns all the unread information from the error/event queue, and removes all the information from the queue. For details of the error messages, refer to page 3–17.

Syntax `:SYSTem:ERRor:ALL?`

Arguments None

Returns `<ecode>,"<edesc>[;<einfo>]"{,<ecode>,"<edesc>[;<einfo>]"}`

Where
`<ecode>::=<NR1>` is the error/event code (–32768 to 32767).
`<edesc>::=<string>` is the description on the error/event.
`<einfo>::=<string>` is the detail of the error/event.

Measurement Modes All

Examples `:SYSTem:ERRor:ALL?`
might return
`-130, "Suffix error; Unrecognized suffix, INPut:MLEVel -10dB"`,
indicating that the unit of the reference level is improper.

:SYSTem:ERRor:CODE:ALL? (Query Only)

Returns all the unread error/event codes from the error/event queue, and removes all the information from the queue. For details of the error messages, refer to page 3–17.

Syntax `:SYSTem:ERRor:CODE:ALL?`

Arguments None

Returns `<ecode>{,<ecode>}`

Where
`<ecode>::=<NR1>` is the error/event code, ranging from –32768 to 32767.

Measurement Modes All

Examples `:SYSTem:ERRor:CODE:ALL?`
might return `-101,-108` of the error codes.

:SYSTem:ERRor:CODE[:NEXT]? (Query Only)

Returns the most recent unread error/event code from the error/event queue, and removes that information from the queue. For details of the error messages, refer to page 3–17.

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

Arguments None

Returns `<ecode>::=<NR1>` is the error/event code, ranging from –32768 to 32767.

Measurement Modes All

Examples `:SYSTem:ERRor:CODE:NEXT?`
might return `-101` of the error code.

:SYSTem:ERRor:COUNt? (Query Only)

Returns the number of unread errors/events placed in the error/event queue.

Syntax `:SYSTem:ERRor:COUNt?`

Arguments None

Returns `<enum>::=<NR1>` is the number of errors/events.

Measurement Modes All

Examples `:SYSTem:ERRor:COUNt?`
might return 2, indicating that the error/event queue contains two of unread errors/events.

:SYSTem:ERRor[:NEXT]? (Query Only)

Returns the next item from the error/event queue, and removes that item from the queue. For details of the error messages, refer to page 3–17.

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

Arguments None

Returns `<ecode>,"<edesc>[;<einfo>]"`

Where
`<ecode>::=<NR1>` is the error/event code, ranging from –32768 to 32767.
`<edesc>::=<string>` is the description on the error/event.
`<einfo>::=<string>` is the detail of the error/event.

Measurement Modes All

Examples `:SYSTem:ERRor:NEXT?`
might return
`-130, "Suffix error; Unrecognized suffix, INPut:MLEVel -10dB"`,
indicating that the unit is improper.

:SYSTem:KLOCk (?)

Determines whether to lock or unlock the front panel key controls.

Syntax

```
:SYSTem:KLOCk { OFF | ON | 0 | 1 }
:SYSTem:KLOCk?
```

Arguments

`OFF` or `0` unlocks the front panel key controls.

`ON` or `1` locks the front panel key controls.

Measurement Modes

All

Examples

```
:SYSTem:KLOCk ON
```

locks the front panel key controls.

:SYSTem:OPTions? (Query Only)

Queries the options installed in the analyzer.
This command is equivalent to the IEEE common command *OPT?.

Syntax

```
:SYSTem:OPTions?
```

Arguments

None

Returns

`<option>::=<string>` contains the comma-separated option numbers.

Measurement Modes

All

Examples

```
:SYSTem:OPTions?
```

might return `"05,10,12"`, indicating that Option 05, 10, and 12 are currently installed in the analyzer.

Related Commands

`:INSTrument[:SELect]`

:SYSTem:PRESet (No Query Form)

Restores the analyzer to the defaults.
This command is equivalent to the PRESET key on the front panel.

Syntax `:SYSTem:PRESet`

Arguments None

Measurement Modes All

Examples `:SYSTem:PRESet`
restores the analyzer to the defaults.

:SYSTem:TIME (?)

Sets or queries the time (hours, minutes, and seconds). This command is equivalent to the time setting through the Windows Control Panel.

Syntax `:SYSTem:TIME <hour>,<minute>,<second>`

`:SYSTem:TIME?`

Arguments `<hour>::= <NRf>` specifies the hours. Range: 0 to 23.

`<minute>::=<NRf>` specifies the minutes. Range: 0 to 59.

`<second>::=<NRf>` specifies the seconds. Range: 0 to 59.

These values are rounded to the nearest integer.
*RST has no effect on the settings.

Measurement Modes All

Examples `:SYSTem:TIME 10,15,30`
sets the time to 10:15:30.

Related Commands `:INSTrument[:SELect]`

:SYSTem:VERSion? (Query Only)

Returns the SCPI version number for which the analyzer complies.

Syntax `:SYSTem:VERSion?`

Arguments None

Returns <NR2> has the form YYYY.V where the Ys represent the year-version (for example, 1999) and the V represents an approved revision number for that year.

Measurement Modes All

Examples `:SYSTem:VERSion?`
might return `1999.0` for the SCPI version.

:TRACe Commands

The :TRACe commands set up display of Trace 1 and 2.

***NOTE**. The :TRACe commands are available in the S/A (spectrum analysis) mode except real-time. To use a command in this group, you must have selected SANORMAL or SASGRAM with the :INSTrument [:SELect] command.*

Command Tree

Header	Parameter
:TRACe<x> | :DATA<x>	
:AVERage	
:CLEar	
:COUNt	<numeric_value>
:DDETector	MAXimum | MINimum | PTPeak
:MODE	NORMal | AVERage | MAXHold | MINHold | FREeze | OFF

Where
TRACe<x>::={ TRACe[1] | TRACe2 } or DATA<x>::={ DATA[1] | DATA2 }

TRACe[1] or DATA[1] indicates that this setup is made for Trace 1.

TRACe2 or DATA2 indicates that this setup is made for Trace 2.

:TRACe<x>|:DATA<x>:AVERage:CLEar (No Query Form)

Clears average data and counter, and restarts the average process for the specified trace.

This command is effective when you select AVERage, MAXHold or MINHold with the :TRACe<x>|:DATA<x>:MODE command.

Syntax :TRACe<x>|:DATA<x>:AVERage:CLEar

Arguments None

Measurement Modes SANORMAL, SASGRAM

Examples :TRACe1:AVERage:CLEar
clears average data and counter, and restarts the average process for Trace 1.

Related Commands :TRACe<x>|:DATA<x>:MODE

:TRACe<x>|:DATA<x>:AVERage:COUNt (?)

Sets or queries the number of traces to combine using the :MODE setting (refer to page 2–334).

This command is effective when you select AVERage, MAXHold or MINHold with the :TRACe<x>|:DATA<x>:MODE command.

Syntax :TRACe<x>|:DATA<x>:AVERage:COUNt <value>

:TRACe<x>|:DATA<x>:AVERage:COUNt?

Arguments <value>::=<NR1> specifies the number of traces to combine for averaging. Range: 1 to 100000 (default: 20)

Measurement Modes SANORMAL, SASGRAM

Examples :TRACe1:AVERage:COUNt 64
sets the average count to 64 for Trace 1.

Related Commands :TRACe<x>|:DATA<x>:MODE

:TRACe<x>|:DATA<x>:DDETector (?)

Selects or queries the display detector (method to be used for decimating traces to fit the available horizontal space on screen).

The number of horizontal pixels on screen is generally smaller than that of waveform data points. When actually displayed, the waveform data is therefore thinned out, according to the number of pixels, for being compressed. For the details, refer to the *RSA2203A and RSA2208A User Manual*.

Syntax

```
:TRACe<x>|:DATA<x>:DDETector { MAXimum | MINimum | PTPeak }
:TRACe<x>|:DATA<x>:DDETector?
```

Arguments

MAXimum displays the maximum data value for each pixel.

MINimum displays the minimum data value for each pixel.

PTPeak displays the maximum and minimum data value by connecting them with a line for each pixel.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:TRACe1:DDETector MAXimum
```
displays the maximum data value for each pixel on Trace 1.

:TRACe<x>|:DATA<x>:MODE (?)

Selects or queries how to display Trace 1 and/or Trace 2.

Syntax

```
:TRACe<x>|:DATA<x>:MODE { NORMal | AVERage | MAXHold | MINHold |
FREeze | OFF }
```

```
:TRACe<x>|:DATA<x>:MODE?
```

Arguments

NORMal selects an ordinary spectrum display.

AVERage displays averaged waveform of the specified trace. The number of averages is set with the :TRACe<x>|:DATA<x>:AVERage:COUNt command.

MAXHold holds the maximum level at each frequency.

MINHold holds the minimum level at each frequency.

FREeze stops updating the display. But the data acquisition and measurement continues.

OFF displays no trace.

Measurement Modes

SANORMAL, SASGRAM

Examples

```
:TRACe1:MODE AVERage
```

displays averaged waveform of Trace 1.

Related Commands

:TRACe<x>|:DATA<x>:AVERage:COUNt

:TRIGger Commands

The :TRIGger commands control triggering.
For details on the trigger, refer to the *RSA2203A and RSA2208A User Manual*.

Command Tree

Header	Parameter
:TRIGger	
[:SEQuence]	
:LEVel	
:IF	<numeric_value>
:MODE	AUTO | NORMal
:MPOSition?	<numeric_value>
:OPOSition?	<numeric_value>
:POSition	<numeric_value>
:SAVE	
:COUNt	
:MAXimum	<numeric_value>
[:STATe]	<boolean>
[:STATe]	<boolean>
:SLOPe	POSitive | NEGative | PNEGative | NPOSitive
:SOURce	IF | EXTernal

:TRIGger[:SEQuence]:LEVel:IF (?)

Sets or queries the trigger level when you select IF with the :TRIGger[:SEQuence]:SOURce command.

Syntax

:TRIGger[:SEQuence]:LEVel:IF <value>

:TRIGger[:SEQuence]:LEVel:IF?

Arguments

<value>::=<NR1> specifies the IF trigger level. Range: 1 to 100%.

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

:TRIGger:SEQuence:LEVel:IF 50pct
sets the IF trigger level to 50%.

Related Commands

:TRIGger[:SEQuence]:SOURce

:TRIGger[:SEQuence]:MODE (?)

Selects or queries the trigger mode.

Syntax

```
:TRIGger[:SEQuence]:MODE { AUTO | NORMal }
:TRIGger[:SEQuence]:MODE?
```

Arguments

AUTO generates a trigger when the :INITiate[:IMMediate] command is sent. In the single mode, data for one waveform is acquired and displayed. In the continuous mode, data acquisition and display are repeated.

NORMal specifies that when the :INITiate[:IMMediate] command is sent after trigger conditions have been preset, the trigger occurs before the process stops. You can set the trigger source, slope, level, and position as the trigger conditions.

***NOTE**. When you select Auto for the trigger mode, you cannot set the trigger source, slope, position, and level.*

At *RST, the trigger mode is set to Auto.

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:TRIGger:SEQuence:MODE AUTO
```

selects the auto trigger.

Related Commands

:INITiate:CONTinuous, :INITiate[:IMMediate]

:TRIGger[:SEQuence]:MPOSition? (Query Only)

Queries the trigger occurence point in one block data acquired on the memory when measurement results are obtained with the :FETCh or :READ commands.

Syntax `:TRIGger[:SEQuence]:MPOSition? <value>`

Arguments `<value>::=<NR1>` specifies the block number. Zero indicates the latest block. Range: –499 to 0

Returns `<NR1>` represents the trigger occurence point. The returned value depends on whether a trigger occurred or not, as shown in the table below.

Trigger occurrence	Returned value [1]
Trigger occurred	–1024 to (block size) × 1024 –1
No trigger occurred	(block size) × 1024

[1] **The block size is set with [:SENSe]:BSIZe.**

A minus value indicates that the trigger occurred before the block data acquisition.

If you send :TRIGger[:SEQuence]:MPOSition? MINimum | MAXimum when the measurement is not performed, "Execution error" (–200) is returned.

***NOTE**. When you select PNEGative or NPOSitive with the :TRIGger[:SEQuence] :SLOPe command, the returned value is the same as the :TRIGger[:SEQuence] :OPOSition? query command because the analyzer cannot determine the trigger occurrence point.*

Measurement Modes `SARTIME`, `SAZRTIME`, `DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples
```
:TRIGger:SEQuence:MPOSition?
```
might return 123, indicating that the trigger occurred at the 123th data point in the block.

Related Commands `[:SENSe]:BSIZe`, `:TRIGger[:SEQuence]:OPOSition?`, `:TRIGger[:SEQuence]:SLOPe`

:TRIGger[:SEQuence]:OPOSition? (Query Only)

Queries the trigger output point in one block data acquired when measurement results are obtained with the :FETCh or :READ commands (the trigger output point is indicated by "T" in the overview on screen).

Syntax

```
:TRIGger[:SEQuence]:OPOSition? <value>
```

Arguments

<value>::=<NR1> specifies the block number. Zero indicates the latest block. Range: –499 to 0

Returns

<NR1> represents the trigger output point. The value depends on whether a trigger occurred or not, as shown in the table below.

Trigger occurrence	Returned value [1]
Trigger occurred	–1024 to (block size) × 1024 –1
No trigger occurred	(block size) × 1024

1 The block size is set with [:SENSe]:BSIZe.

A minus value indicates that the trigger was output before the block data acquisition.

If you send :TRIGger[:SEQuence]:OPOSition? MINimum | MAXimum when the measurement is not performed, "Execution error" (–200) is returned.

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:TRIGger:SEQuence:OPOSition?
```

might return 134, indicating that the trigger output occurs at the 134th data point in the block.

Related Commands

[:SENSe]:BSIZe

:TRIGger[:SEQuence]:POSition (?)

Sets or queries a trigger position.

Syntax

```
:TRIGger[:SEQuence]:POSition <value>
:TRIGger[:SEQuence]:POSition?
```

Arguments

<value>::=<NRf> specifies the trigger position. Range: 0 to 100%. The trigger position is represented in percentage within a block. For example, 50% specifies that the trigger will occur at the middle frame in a block.

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:TRIGger:SEQuence:POSition 10pct
```

sets the trigger position to 10%.

:TRIGger[:SEQuence]:SAVE:COUNt[:STATe](?)

Selects whether or not to set a limit on the number of times that data is saved.

Syntax

```
:TRIGger[:SEQuence]:SAVE:COUNt[:STATe] { OFF | ON | 0 | 1 }
:TRIGger[:SEQuence]:SAVE:COUNt[:STATe]?
```

Arguments

OFF or 0 specifies that no limit on data save operations is set. In this case, data saving is halted using the **RUN/STOP** key on the front panel or the :ABORt or :INITiate command.

ON or 1 specifies that data saving is halted when the number of data save operations reaches the limit set by the :TRIGger[:SEQuence]:SAVE:COUNt :MAXimum command.

***NOTE**. When the internal hard disk becomes full, data saving is halted and the "Media full" error message appears.*

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:TRIGger:SEQuence:SAVE:COUNt:STATe ON
```

specifies that data saving is halted when the number of data save operations reaches the limit.

Related Commands

:ABORt, :INITiate, :TRIGger[:SEQuence]:SAVE:COUNt:MAXimum

:TRIGger[:SEQuence]:SAVE:COUNt:MAXimum(?)

Sets or querie a limit on the number of times that data is saved when :TRIGger [:SEQuence]:SAVE:COUNt[:STATe] is set to On.

Syntax

```
:TRIGger[:SEQuence]:SAVE:COUNt:MAXimum <value>
:TRIGger[:SEQuence]:SAVE:COUNt:MAXimum?
```

Arguments

<value>::=<NR1> specifies a limit on the number of times that data is saved. Range: 1 to 16383.

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:TRIGger:SEQuence:SAVE:COUNt:MAXimum 10000
```

sets the limit to 10000.

Related Commands

:TRIGger[:SEQuence]:SAVE:COUNt[:STATe]

:TRIGger[:SEQuence]:SAVE[:STATe](?)

Determines whether to enable or disable the Save-on-Trigger function (saves one block of input data to the .IQT file each time a trigger occurs).

Syntax

```
:TRIGger[:SEQuence]:SAVE[:STATe] { OFF | ON | 0 | 1 }
:TRIGger[:SEQuence]:SAVE[:STATe]?
```

Arguments

OFF or 0 disables the Save-on-Trigger (default).

ON or 1 enables the Save-on-Trigger.

Measurement Modes

SARTIME, SAZRTIME, DEMADEM, TIMCCDF, TIMTRAN, TIMPULSE

Examples

```
:TRIGger:SEQuence:SAVE:STATe ON
```

enables the Save-on-Trigger function.

Related Commands

:TRIGger[:SEQuence]:SAVE:COUNt[:STATe]

:TRIGger[:SEQuence]:SLOPe (?)

Selects or queries the trigger slope.

Syntax

```
:TRIGger[:SEQuence]:SLOPe { POSitive | NEGative | PNEGative
| NPOSitive }
```

```
:TRIGger[:SEQuence]:SLOPe?
```

Arguments

`POSitive` generates a trigger on the rising edge of the trigger signal.

`NEGative` generates a trigger on the falling edge of the trigger signal.

`PNEGative` specifies that the data of the first block is acquired by generating the trigger on the rising edge of the trigger signal. The data of the next block is acquired by generating the trigger on the falling edge of the trigger signal. The rising and falling edges are changed alternately each time acquisition of one-block data is completed.

`NPOSitive` specifies that the data of the first block is acquired by generating the trigger on the falling edge of the trigger signal. The data of the next block is acquired by generating the trigger on the rising edge of the trigger signal. The rising and falling edges are changed alternately each time acquisition of one-block data is completed.

Measurement Modes

`SARTIME`, `SAZRTIME`, `DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples

```
:TRIGger:SEQuence:SLOPe POSitive
```

generates a trigger on the rising edge of the trigger signal.

:TRIGger[:SEQuence]:SOURce (?)

Selects or queries the trigger source.

Syntax

```
:TRIGger[:SEQuence]:SOURce { IF | EXTernal }
:TRIGger[:SEQuence]:SOURce?
```

Arguments

`IF` defines the internal IF (Intermediate Frequency) signal as the trigger source (default).

`EXTernal` defines as the trigger source, the external signal that is input through the TRIG IN connector on the rear panel. The trigger level is fixed internally. Refer to the *RSA2203A and RSA2208A User Manual* for the external trigger level specification.

Measurement Modes

`SARTIME`, `SAZRTIME`, `DEMADEM`, `TIMCCDF`, `TIMTRAN`, `TIMPULSE`

Examples

```
:TRIGger:SEQuence:SOURce IF
```

selects the IF trigger.

:UNIT Commands

The :UNIT commands specify fundamental units for measurement.

Command Tree

Header	Parameter
:UNIT	
:ANGLe	DEG | RAD

:UNIT:ANGLe (?)

Specifies or queries the fundamental unit of angle.

Syntax

```
:UNIT:ANGLe { DEG | RAD }
:UNIT:ANGLe?
```

Arguments

DEG selects degree as the unit of angle.

RAD selects radian as the unit of angle.

Measurement Modes

All

Examples

```
:UNIT:ANGLe RAD
```

selects radian as the unit of angle.

Retrieving Response Message

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

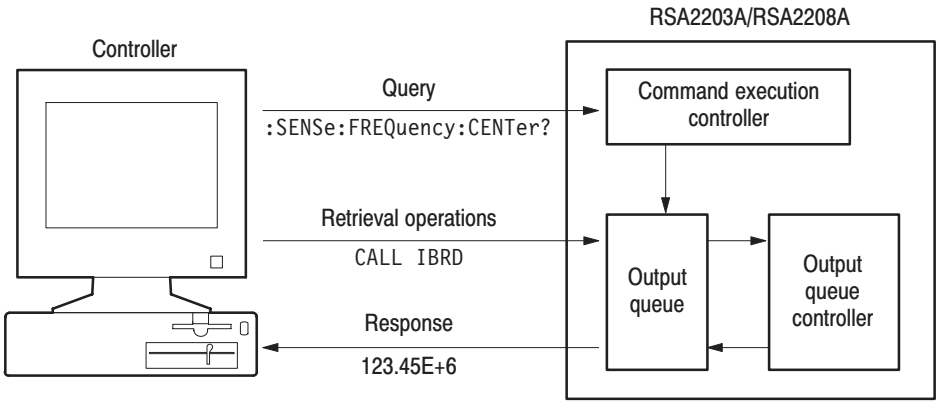


Figure 2–24: Retrieving response message

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

You can use the MAV bit of the Status Byte Register (SBR) to check whether the Output Queue contains a response message. For details, refer to *Status Byte Register (SBR)* on page 3–6.

Status and Events

Status and Events

The SCPI interface in the analyzer includes a status and event reporting system that enables the user to monitor crucial events that occur in the instrument. The analyzer is equipped with four registers and one queue that conform to IEEE Std 488.2-1987. This section will discuss these registers and queues along with status and event processing.

Status and Event Reporting System

Figure 3–1 outlines the status and event reporting mechanism offered in the RSA2200 Series analyzers.

The status and event reporting mechanism contains three major blocks:

- Standard Event Status
- Operation Status
- Questionable Status

The processes performed in these blocks are summarized in the status bytes. They provide the error and event information.

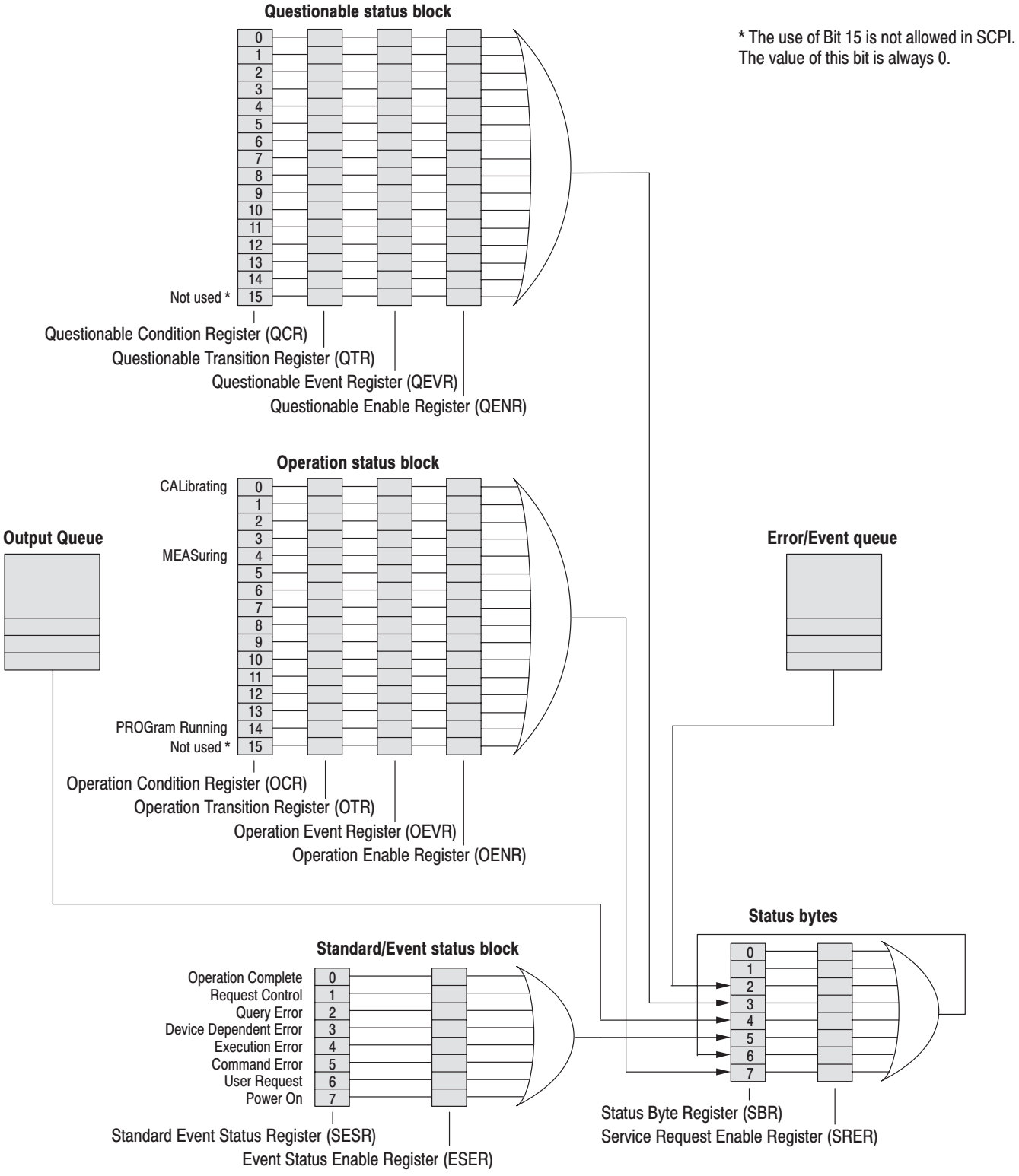


Figure 3–1: Status/Event reporting mechanism

Standard Event Status Block

Reports the power on/off state, command errors, and the running state.

See the Standard/Event Status Block section at the bottom of Figure 3–1. This block contains two registers:

- **Standard Event Status Register (SESR)**
  Consists of eight bits. When an error or another event occurs in the analyzer, the corresponding bit of this register is set. The user cannot write any data in this register.
- **Event Status Enable Register (ESER)**
  Consists of eight bits, and masks the SESR. The mask is user-definable. By obtaining the logical product with SESR, this register can determine whether to set the Event Status Bit (ESB) of the Status Byte Register (SBR).

Processing Flow. When an event occurs, the SESR bit corresponding to the event is set, resulting in the event being stacked in the Error/Event Queue. The SBR OAV bit is also set. If the bit corresponding to the event has also been set in the ESER, the SBR ESB bit is also set.

When a message is sent to the Output Queue, the SBR MAV bit is set.

Operation Status Block

Reports the active state of the function.

See the Operation Status Block section at the middle of Figure 3–1. This block contains four registers:

- **Operation Condition Register (OCR)**
  When the analyzer enters a certain state, the corresponding bit is set. The user cannot write any data in this register.
- **Operation Transition Register (OTR)**
  There are two OTR types:
  - **Operation Positive Transition Register (OPTR)**
    Filters when the bit corresponding to the OCR changes from False (reset) to True (set).
  - **Operation Negative Transition Register (ONTR)**
    Filters when the bit corresponding to the OCR changes from True to False.
- **Operation Event Register (OEVR)**
  In the OEVR, the corresponding bit is set through the OTR filter.
- **Operation Enable Register (OENR)**
  Masks the OEVR. The mask is user-definable. By obtaining the logical product with SBR, this register can determine whether to set the Operation Status Bit (OSB) of the Status Byte Register (SBR).

For the contents of the bits of this register, refer to *Registers* on page 3–5.

Processing Flow. When the specified state changes in the OCR, its bit is set or reset. This change is filtered with a transition register, and the corresponding bit of the OEVR is set. If the bit corresponding to the event has also been set in the OENR, the SBR OSS bit is also set.

Questionable Status Block

Reports the states related to signals and data, for example, the signal generated by the analyzer or the precision of the data to be acquired. The register organization and the processing flow are the same as the Operation Status Block, except that the corresponding bit of the SBR is the QSB.

***NOTE**. The Questionable Status Block is not used in the RSA2203A/RSA2208A analyzer. Any of the values of the registers in this block are always 0.*

Registers

There are three main types of registers:

- Status Registers: stores data relating to instrument status. This register is set by the analyzer.
- Enable Registers: determines whether to set events that occur in the analyzer to the appropriate bit in the status registers and event queues. This type of register can be set by the user.
- Transition Registers: operates as a filter that examines whether an event has occurred or disappeared. This type of register can be set by the user.

Status Registers

There are six status register types:

- Status Byte Register (SBR)
- Standard Event Status Register (SESR)
- Operation Condition Register (OCR)
- Operation Event Register (OEVR)
- Questionable Condition Register (QCR)
- Questionable Event Register (QEVR)

If you need to examine the error or the state of the analyzer, read the contents of these registers.

Status Byte Register (SBR)

The SBR is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2-1987 (see Figure 3–2 and Table 3–1). These bits are used to monitor the output queue, SESR and service requests, respectively. The contents of this register are returned when the *STB? query is used.

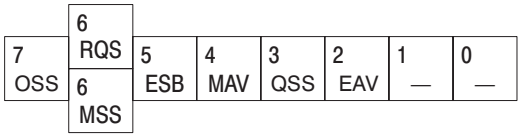


Figure 3–2: The Status Byte Register (SBR)

Table 3–1: SRB bit functions

Bit	Function
7	Operation Summary Status (OSS). Summary of the operation status register.
6	Request Service (RQS)/Master Status Summary (MSS). When the instrument is accessed using the GPIB serial poll command, this bit is called the Request Service (RQS) bit and indicates to the controller that a service request has occurred (in other words, that the GPIB bus SRQ line is LOW). The RQS bit is cleared when serial poll ends. When the instrument is accessed using the *STB? query, this bit is called the Master Status Summary (MSS) bit and indicates that the instrument has issued a service request for one or more reasons. The MSS bit is never cleared to 0 by the *STB? query.
5	Event Status Bit (ESB). This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed.
4	Message Available Bit (MAV). This bit indicates that a message has been placed in the output queue and can be retrieved.
3	Questionable Summary Status (QSS). Summary of the Questionable Status Byte register. It is always zero in the RSA2203A/RSA2208A analyzer.
2	Event Quantity Available (EAV). Summary of the Error Event Queue.
1–0	Not used

Standard Event Status Register (SESR)

The SESR is made up of 8 bits. Each bit records the occurrence of a different type of event, as shown in Figure 3–3 and Table 3–2. The contents of this register are returned when the *ESR? query is used.

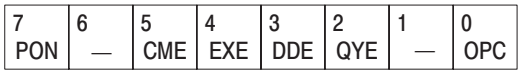


Figure 3–3: The Standard Event Status Register (SESR)

Table 3–2: SESR bit functions

Bit	Function
7	Power On (PON). Indicates that the power to the instrument is on.
6	Not used.
5	Command Error (CME). Indicates that a command error has occurred while parsing by the command parser was in progress.
4	Execution Error (EXE). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons: ■ When a value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument ■ When the command could not be executed properly because the conditions for execution differed from those essentially required
3	Device-Specific Error (DDE). An instrument error has been detected.
2	Query Error (QYE). Indicates that a query error has been detected by the output queue controller. Query errors occur for one of the following reasons: ■ An attempt was made to retrieve messages from the output queue, despite the fact that the output queue is empty or in pending status. ■ The output queue messages have been cleared despite the fact that they have not been retrieved.
1	Not used.
0	Operation Complete (OPC). This bit is set with the results of the execution of the *OPC command. It indicates that all pending operations have been completed.

Operation Condition Register (OCR)

The OCR is made up of 16 bits, which record the occurrence of three types of events, shown in Figure 3–4 and Table 3–3.

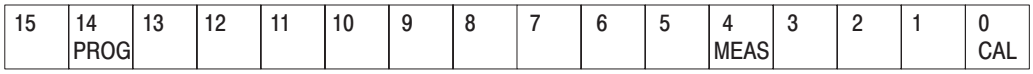


Figure 3–4: The Operation Condition Register (OCR)

Table 3–3: OCR bit functions

Bit	Function
15	Not used.
14	Program Running Bit (PROG): Indicates whether the macro program is in execution. Set while the macro program is run by a :PROGram:EXECute command. Reset when it ends.
13–5	Not used.
4	Measuring Bit (MEAS): Indicates whether the analyzer is in measurement. When the measurement ends after this bit is set in measurement, it is reset. "In measurement" means that one of the following commands is in execution: :INITiate commands :READ commands [:SENSe]:ADEMod[:IMMediate] [:SENSe]:TRANsient[:IMMediate]
3–1	Not used.
0	Calibration Bit (CAL): Indicates whether the analyzer is in measurement. When the measurement ends after this bit is set in calibration, it is reset.

Operation Event Register (OEVR)

In this instrument, this register has the same content as the Operation Condition Register (OCR), described above.

Questionable Condition Register (QCR)

The QCR is not used in the RSA2203A/RSA2208A analyzer.

Questionable Event Register (QEVR)

The QEVR is not used in the RSA2203A/RSA2208A analyzer.

Enable Registers

There are four enable register types:

- Event Status Enable Register (ESER)
- Service Request Enable Register (SRER)
- Operation Enable Register (OENR)
- Questionable Enable Register (QENR)

Each bit in these enable registers corresponds to a bit in the controlling status register. By setting and resetting the bits in the enable register, the user can determine whether or not events that occur will be registered to the status register and queue.

Event Status Enable Register (ESER)

The ESER is made up of bits defined exactly the same as bits 0 through 7 in the SESR (see Figure 3–5). This register is used by the user to designate whether the SBR ESB bit should be set when an event has occurred and whether the corresponding SESR bit has been set.

To set the SBR ESB bit (when the SESR bit has been set), set the ESER bit corresponding to that event. To prevent the ESB bit from being set, reset the ESER bit corresponding to that event.

Use the *ESE command to set the bits of the ESER. Use the *ESE? query to read the contents of the ESER.

7	6	5	4	3	2	1	0
PON	—	CME	EXE	DDE	QYE	—	OPC

Figure 3–5: The Event Status Enable Register (ESER)

Service Request Enable Register (SRER)

The SRER is made up of bits defined exactly the same as bits 0 through 7 in the SBR (see Figure 3–6). This register is used by the user to determine what events will generate service requests.

The SRER bit 6 cannot be set. Also, the RQS is not maskable.

The generation of a service request with the GPIB interface involves changing the SRQ line to LOW and making a service request to the controller. The result is that a status byte for which an RQS has been set is returned in response to serial polling by the controller.

Use the *SRE command to set the bits of the SRER. Use the *SRE? query to read the contents of the SRER. Bit 6 must normally be set to 0.

7	6	5	4	3	2	1	0
OSB	—	ESB	MAV	QSB	—	—	—

Figure 3–6: The Service Request Enable Register (SRER)

Operation Enable Register (OENR)

Consists of the bits that are defined as the same contents as bits 0 to 15 of the OEVR. This register is used to specify whether to set the SBR OSB bit when occurrence of an event sets the corresponding OEVR bit.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG										MEAS				CAL

Figure 3–7: Operation Enable Register (OENR)

To set the contents of the OENR, use a :STATus:OPERation:ENABle command. To query its contents, use query command STATus:OPERation:ENABle?.

Questionable Enable Register (QENR)

The QENR is not used in the RSA2203A/RSA2208A analyzer.

Transition Registers

There are two transition register types:

- Operation Transition Register (OTR)
- Questionable Transition Register (QTR)

Operation Transition Register (OTR)

Consists of the bits that are defined as the same contents as bits 0 to 15 of the OCR (refer to page 3–9). This bit has two functions. One is positive transition filtering, which filters when the corresponding bit of the OCR changes from False (reset) to True (set). The other is negative transition filtering, which filters when this bit changes from True to False.

To set the OTR bit to use the register as the positive transition filter, use a :STATus:OPERation:PTRansition command. To read the contents from it, use query :STATus:OPERation:PTRansition?.

To set the OTR bit to use the register as the negative transition filter, use a :STATus:OPERation:NTRansition command. To read the contents from it, use query :STATus:OPERation:NTRansition?.

15	14 PROG	13	12	11	10	9	8	7	6	5	4 MEAS	3	2	1	0 CAL

Figure 3–8: Operation Transition Register (OTR)

Questionable Transition Register (QTR)

The QTR is not used in the RSA2203A/RSA2208A analyzer.

Queues

There are two types of queues in the status reporting system used in the analyzer: output queues and event queues.

Output Queue

The output queue is a FIFO queue and holds response messages to queries, where they await retrieval. When there are messages in the queue, the SBR MAV bit is set.

The output queue will be emptied each time a command or query is received, so the controller must read the output queue before the next command or query is issued. If this is not done, an error will occur and the output queue will be emptied; however, the operation will proceed even if an error occurs.

Event Queue

The event queue is a FIFO queue and stores events as they occur in the analyzer. If more than 32 events occur, event 32 will be replaced with event code –350 (“Queue Overflow”). The error code and text are retrieved using the :SYSTem:ERRor queries.

Status and Event Processing Sequence

Figure 3–9 shows an outline of the sequence for status and event processing.

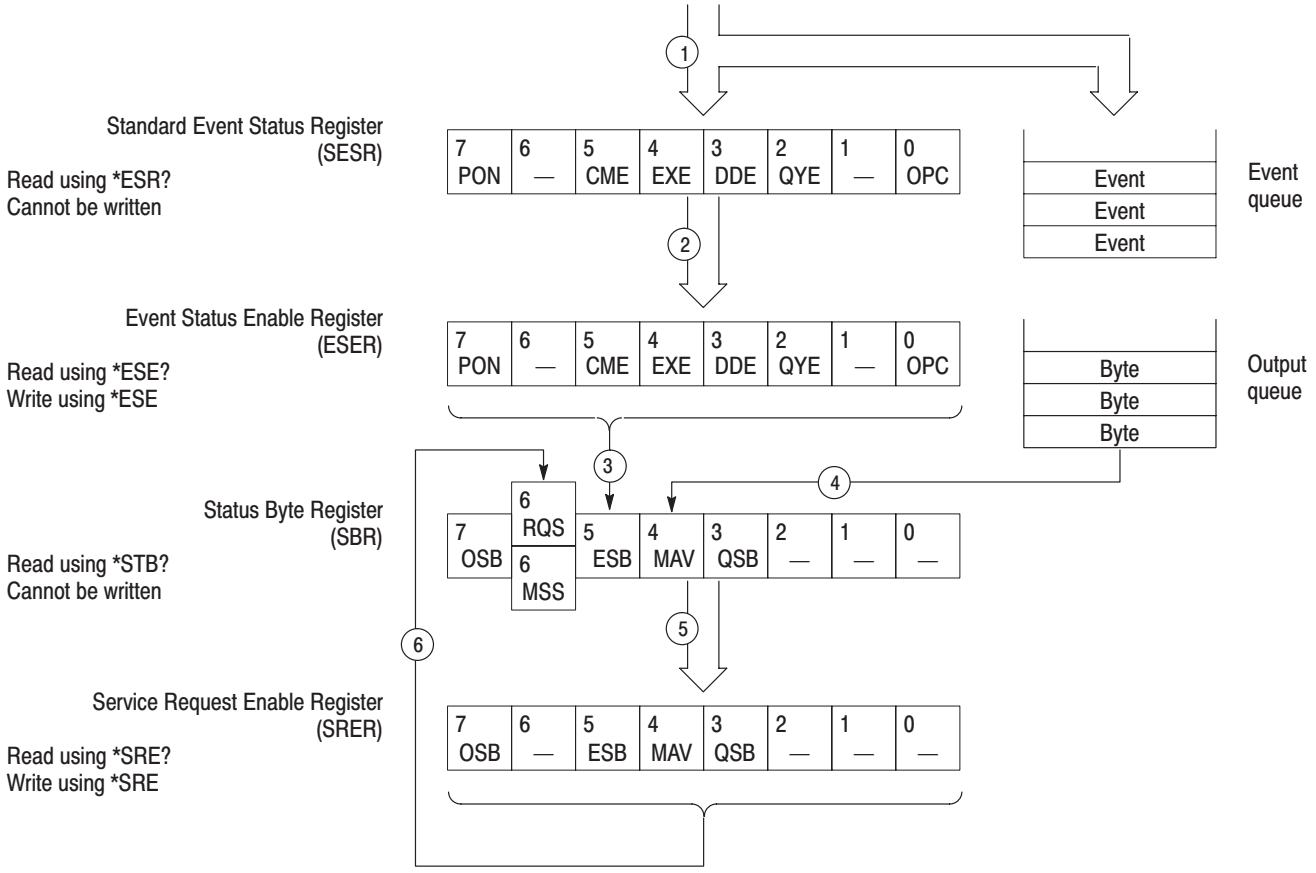


Figure 3–9: Status and event processing sequence

1. If an event has occurred, the SESR bit corresponding to that event is set and the event is placed in the event queue.
2. A bit corresponding to that event in the ESER has is set.
3. The SBR ESB bit is set to reflect the status of the ESER.
4. When a message is sent to the output queue, the SBR MAV bit is set.
5. Setting either the ESB or MAV bits in the SBR sets the respective bit in the SRER.
6. When the SRER bit is set, the SBR MSS bit is set and a service request is generated when using the GPIB interface.

Synchronizing Execution

Almost all commands are executed in the order in which they are sent from the controller, and the execution of each command is completed in a short period of time. However, the following commands perform data analysis in another thread, and another command can thus be executed concurrently:

:INITiate commands
:PROGram[:SELected]:EXEcute
:PROGram[:SELected]:NAME
:READ commands
[:SENSe]:ADEMod[:IMMediate]
[:SENSe]:TRANsient[:IMMediate]

These commands are designed so that the next command to be sent is executed without waiting for the previous command to be completed. In some cases, a process executed by another command must first be completed before these commands can be executed; in other cases, these commands must be completed before the next command is executed.

You have two options to achieve command synchronization:

- Using the status and event reporting function
- Using synchronizing commands

Using the Status and Event Reporting Function

In the following example, a :READ command is used to obtain the measurement results while the Operation Condition Register (OCR) is being used to provide synchronization.

```
:STATus:OPERation:NTRansition 16
                // Set the filter of the OCR MEASuring bit
:STATus:OPERation:ENABle 16
                // Enable the filter of the OCR MEASuring bit
*SRE 128        // Set the SRER OSS bit
:READ:SPECtrum?   // Obtain the measurement results
```

The command waits for generation of SRQ.

Using Synchronizing Commands

The IEEE-488.2 common commands include the following synchronizing commands:

```
*OPC
*OPC?
*WAI
```

Using the *OPC Command. The *OPC command sets the SESR OPC bit when all the operations for which it is waiting are completed. If the GPIB interface is in use, you can synchronize the execution by using this command together with the serial polling or service request function.

The following is a command sequence example:

```
*ESE 1          // Enable the ESER OPC bit
*SRE 32         // Enable the SRER ESB bit
:ABORt;INITiate:IMMediate;*OPC
                // Wait for SRQ to provide synchronization
```

Using the Query *OPC? The query *OPC? writes ASCII code “1” into the Output Queue when all operations for which it is waiting are completed. You can provide synchronization using the command string as the following example:

```
:ABORt;INITiate:IMMediate;*OPC?
```

The command waits until “1” is written into the Output Queue. When the command goes to the Output Queue to read the data, a time-out may occur before the data is written into the queue.

Using the *WAI Command. After the process of the preceding command is completed, the *WAI command begins to execute the process of the next command as the following example:.

```
:ABORt;INITiate:IMMediate;*WAI
                // Wait for the *WAI process to provide synchronization
```

Error Messages and Codes

Tables 3–4 through 3–7 show the SCPI standard error codes and messages used in the status and event reporting system in the analyzer.

Event codes and messages can be obtained by using the queries :SYSTem:ERRor. These are returned in the following format:

```
<event code>,"<event message>"
```

Command Errors

Command errors are returned when there is a syntax error in the command.

Table 3–4: Command errors

Error code	Error message
–100	command error
–101	invalid character
–102	syntax error
–103	invalid separator
–104	data type error
–105	GET not allowed
–108	parameter not allowed
–109	missing parameter
–110	command header error
–111	header separator error
–112	program mnemonic too long
–113	undefined header
–114	header suffix out of range
–120	numeric data error
–121	character
–123	exponent too large
–124	too many digits
–128	numeric data not allowed
–130	suffix error
–131	invalid suffix
–134	suffix too long
–138	suffix not allowed
–140	character data error
–141	invalid character data
–144	character data too long
–148	character data not allowed
–150	string data error
–151	invalid string data
–158	string data not allowed

Table 3–4: Command errors (Cont.)

Error code	Error message
–160	block data error
–161	invalid block data
–168	block data not allowed
–170	command expression error
–171	invalid expression
–178	expression data not allowed
–180	macro error
–181	invalid outside macro definition
–183	invalid inside macro definition
–184	macro parameter error

Execution Errors

These error codes are returned when an error is detected while a command is being executed.

Table 3–5: Execution errors

Error code	Error message
–200	execution error
–201	invalid while in local
–202	settings lost due to RTL
–210	trigger error
–211	trigger ignored
–212	arm ignored
–213	init ignored
–214	trigger deadlock
–215	arm deadlock
–220	parameter error
–221	settings conflict
–222	data out of range
–223	too much data
–224	illegal parameter value
–225	out of memory
–226	lists not same length
–230	data corrupt or stale
–231	data questionable
–240	hardware error
–241	hardware missing
–250	mass storage error
–251	missing mass storage
–252	missing media
–253	corrupt media
–254	media full
–255	directory full
–256	FileName not found
–257	FileName error
–258	media protected

Table 3–5: Execution errors (Cont.)

Error code	Error message
–260	execution expression error
–261	math error in expression
–270	execution macro error
–271	macro syntax error
–272	macro execution error
–273	illegal macro label
–274	execution macro parameter error
–275	macro definition too long
–276	macro recursion error
–277	macro redefinition not allowed
–278	macro header not found
–280	program error
–281	cannot create program
–282	illegal program name
–283	illegal variable name
–284	program currently running
–285	program syntax error
–286	program runtime error

Device Specific Errors

These error codes are returned when an internal instrument error is detected. This type of error may indicate a hardware problem.

Table 3–6: Device specific errors

Error code	Error message
–300	device specific error
–310	system error
–311	memory error
–312	PUD memory lost
–313	calibration memory lost
–314	save/recall memory lost
–315	configuration memory lost
–330	self test failed
–350	queue overflow

Query Errors

These error codes are returned in response to an unanswered query.

Table 3–7: Query errors

Error code	Error message
–400	query error
–410	query interrupted
–420	query unterminated
–430	query deadlocked
–440	query unterminated after indefinite period

Programming Examples

Programming Examples

This section shows an application program sample that controls the analyzer through the GPIB and a macro program execution sample that uses :PROGram commands.

- Application program sample
- Macro program execution sample

Application Program Sample

This section shows an application program sample that performs two measurements:

- **Channel power measurement (measCHPOWER() subroutine)**
  In the S/A (spectrum analysis) mode, the *OPC command is used to provide synchronization while channel power measurement is being performed. Then, the measured data is saved in a file.
- **FM signal measurement (measFM() subroutine)**
  In the Demod (modulation analysis) mode, the status byte MAV bit is used to provide synchronization while the FM vector signal measurement is being performed. Then, the measured data is saved in a file.

This program has been scripted for use in Microsoft Visual C++ 6.0. It operates with an IBM PC-compatible system equipped with National Instruments GPIB board and driver software (operation capabilities confirmed with Windows 98 and National Instruments GPIB board PCI-GPIB). To enable this program, the analyzer must have been set to DEV1 by using wibconf or other means.

```
//
// Sample program
//
// Channel power measurement & FM signal measurement
//
#include <windows.h>
#include <stdio.h>
#include <string.h>

#include "decl-32.h"

#define LONG_TIME T100s
#define NORMAL_TIME T10s

#define BOARD_NAME "GPIB0"
#define MAX_BUF (1024)

// Bit definition of SBR (Status Byte Register)
#define ESB (1<<5)        // ESB (Event Status Bit)
#define MAV (1<<4)        // MAV (Message Available)
#define EAV (1<<2)        // EAV (Event Queue Available)
```

```
char readBuf[MAX_BUF + 1];
char openDevice [MAX_BUF/2 + 1];

void GpibClose(void);
void GpibError(char *errorMessage);
void GpibExit(int code);
void GpibOpen(char *device);
void GpibRead(char *resp, int count);
void GpibReadFile(char *filename);
int  GpibSerialPoll(void);
void GpibTimeOut(int timeout);
void GpibWait(int wait);
void GpibWrite(char *string);
void measCHPOWER(void);
void measFM(void);
void WaitOPC(void);
void WaitMAV(void);

int GpibDevice;           // Device descriptor
int GpibBoard;            // GPIB board descriptor
int GpibCount;            // Store ibcnt
int GpibStatus;           // Store ibsta


// Main routine
void
main(int argc, char *argv[])
{
   strcpy(openDevice, "dev1");

   GpibOpen(openDevice);  // Detect the specified device

   measCHPOWER();         // Channel power measurement

   measFM();              // FM signal measurement

   GpibClose();           // Terminate the device and board

}
```

```
// Channel power measurement
void
measCHPOWER(void)
{
   GpibWrite("*CLS");     // Clear the status register
   GpibWrite("*ESE 1");   // Set the OPC bit of ESER
   GpibWrite("*SRE 32");  // Set the ESB bit of SRER

   // Set up the analyzer
   GpibTimeOut(LONG_TIME);
   GpibWrite("INSTrument 'SANORMAL'");
   GpibWrite("*RST");     // Reset the analyzer
   GpibTimeOut(NORMAL_TIME);
   GpibWrite("CONFigure:SPECtrum:CHPower");
   GpibWrite("FREQuency:CENTer 1GHz");
   GpibWrite("FREQuency:SPAN 1MHz");
   GpibTimeOut(LONG_TIME);
   GpibWrite("*CAL?");
   GpibRead(readBuf, MAX_BUF);
   printf("*CAL? result = %s\n", readBuf);
   GpibTimeOut(NORMAL_TIME);
   GpibWrite("CHPower:BANDwidth:INTegration 300kHz");
   GpibWrite("SPECtrum:AVERage ON");
   GpibWrite("SPECtrum:AVERage:COUNt 100");

   // Perform the measurement
   GpibTimeOut(LONG_TIME);
   GpibWrite("INITiate:CONTinuous OFF;*OPC");
   WaitOPC();             // Wait for the OPC bit set
   GpibWrite("INITiate;*OPC");
   WaitOPC();
   GpibTimeOut(NORMAL_TIME);

   // Get measurement results and save them to the file chpower
   GpibWrite("FETCh:SPECtrum:CHPower?");
   GpibReadFile("chpower");
}
```

```
// FM signal measurement
void
measFM(void)
{
   // Set up the analyzer
   GpibTimeOut(LONG_TIME);
   GpibWrite("INSTrument 'DEMADEM'");
   GpibWrite("*RST");     // Reset the analyzer
   GpibTimeOut(NORMAL_TIME);
   GpibWrite("CONFigure:ADEMod:FM");
   GpibWrite("FREQuency:CENTer 1GHz");
   GpibWrite("FREQuency:SPAN 1MHz");
   GpibWrite("BSIZe 100");
   GpibTimeOut(LONG_TIME);
   GpibWrite("*CAL?");
   GpibRead(readBuf, MAX_BUF);
   printf("*CAL? result = %s\n", readBuf);
   GpibTimeOut(NORMAL_TIME);
   GpibWrite("ADEMod:LENGth 102400");

   GpibWrite("*CLS");     // Clear the status register
   GpibWrite("*SRE 16");  // Set the MAV bit of SRER

   // Perform the measurement
   GpibTimeOut(LONG_TIME);
   GpibWrite("READ:ADEMod:FM?");
   WaitMAV();             // Wait for the MAV bit set
   GpibTimeOut(NORMAL_TIME);

   // Get measurement results and save them to the file fm
   GpibReadFile("fm");
}
```

```
// Wait for the OPC (Operation complete) bit set
void
WaitOPC(void)
{
   int statusByte;

   // Wait for SRQ
   GpibWait(RQS);
   if (GpibStatus & TIMO)
   {
      fprintf(stderr, "Timeout occurred in waiting
         SRQ cycle.\n");
      GpibExit(0);
   }

   // Serial poll
   statusByte = GpibSerialPoll();
   if (statusByte & ESB)
   {
      printf("ESB bit is TRUE\n");
      GpibWrite("*ESR?");
      GpibRead(readBuf, MAX_BUF);
      printf("Standard Event Status Register = %s\n", readBuf);
   }
   if (statusByte & MAV)
      printf("MAV bit is TRUE\n");
   if (statusByte & EAV)
      printf("EAV bit is TRUE\n");
}
```

```
// Wait for the MAV (Message Available) bit set
void
WaitMAV(void)
{
   int statusByte;

   // Wait for SRQ
   GpibWait(RQS);
   if (GpibStatus & TIMO)
   {
      fprintf(stderr, "Timeout occurred in waiting SRQ
         cycle.\n");
      GpibExit(0);
   }

   // Serial poll
   statusByte = GpibSerialPoll();
   if (statusByte & MAV)
      printf("MAV bit is TRUE\n");
   if (statusByte & EAV)
      printf("EAV bit is TRUE\n");
}
```

```
// Open the GPIB device
void
GpibOpen(char *device)
{
   // Assign ID to the device and interface board,
   // and check on error.
   GpibDevice = ibfind(device);
   if (ibsta & ERR)
   {
      GpibError("ibfind Error: Unable to find device");
      GpibExit(0);
   }
   GpibBoard = ibfind(BOARD_NAME);
   if (ibsta & ERR)
   {
      GpibError("ibfind Error: Unable to find board");
      GpibExit(0);
   }

   // Clear the device and check on error.
   ibclr(GpibDevice);
   if (ibsta & ERR)
   {
      GpibError("ibclr Error: Unable to clear device");
      GpibExit(0);
   }
   ibsre(GpibBoard, 0);
   if (ibsta & ERR)
   {
      GpibError("ibclr Error: Unable to clear board");
      GpibExit(0);
   }

   // Set the timeout to 10 seconds (NORMAL_TIME)
   GpibTimeOut(NORMAL_TIME);
}


// Close the GPIB device
void
GpibClose(void)
{
   // Turn off the device and interface board
   ibonl(GpibDevice, 0);
   ibonl(GpibBoard, 0);
}
```

```
// End the program
void
GpibExit(int code)
{
   GpibClose();
   exit(code);
}


// Send string to the device and wait for the completion
void
GpibWrite(char *string)
{
   int count = strlen(string);

   // Send the string
   ibwrt(GpibDevice, string, count);

   // Determine the I/O completion of ibwrt
   if (ibsta & ERR)
   {
      GpibError("ibwrt I/O Error:");
      GpibExit(0);
   }
   else
   {
      GpibCount = ibcnt;
      GpibStatus = ibsta;
      if (GpibSerialPoll() & EAV)
      {
         ibwrt(GpibDevice, "SYSTem:ERRor:ALL?",
            strlen("SYSTem:ERRor:ALL?"));
         ibrd(GpibDevice, readBuf, MAX_BUF);
      fprintf(stderr, "%s\n", readBuf);
      }
   }
}
```

```
// Read response from the device
void
GpibRead(char *resp, int count)
{
   ibrd(GpibDevice, resp, count);

   if (ibsta & ERR)
   {
      GpibError("ibrd I/O Error:");
      GpibExit(0);
   }
   else
   {
      resp[ibcnt] = '\0';
      GpibCount = ibcnt;
      GpibStatus = ibsta;
   }
}


// Read response from the device and write it to a file
void
GpibReadFile(char *filename)
{
   ibrdf(GpibDevice, filename);

   if (ibsta & ERR)
   {
      GpibError("ibrdf I/O Error:");
      GpibExit(0);
   }
   else
   {
      GpibStatus = ibsta;
   }
}
```

```
// Read the status byte
int
GpibSerialPoll(void)
{
   char poll = 0;

   ibrsp(GpibDevice, &poll);
   if (ibsta & ERR)
   {
      GpibError("ibrsp Error:");
      GpibExit(0);
   }
   else
   {
      GpibStatus = ibsta;
   }

   return poll & 0xff;
}


// Set timeout
void
GpibTimeOut(int timeout)
{
   ibtmo(GpibDevice, timeout);
   if (ibsta & ERR)
   {
      GpibError("ibtmo Error:");
      GpibExit(0);
   }
   else
   {
      GpibStatus = ibsta;
   }
}
```

```
// Wait for the specified event
void
GpibWait(int wait)
{
   ibwait(GpibDevice, wait | TIMO);
   if (ibsta & (ERR | TIMO))
   {
      GpibError("ibwait Error:");
   }
   GpibStatus = ibsta;
}

// Display error message by ibsta
void
GpibError(char *errorMessage)
{
   fprintf (stderr, "%s\n", errorMessage);
   fprintf (stderr, "ibsta=(%X)h <", ibsta);

   if (ibsta & ERR ) fprintf (stderr, " ERR");
   if (ibsta & TIMO) fprintf (stderr, " TIMO");
   if (ibsta & END ) fprintf (stderr, " END");
   if (ibsta & SRQI) fprintf (stderr, " SRQI");
   if (ibsta & RQS ) fprintf (stderr, " RQS");
   if (ibsta & CMPL) fprintf (stderr, " CMPL");
   if (ibsta & LOK ) fprintf (stderr, " LOK");
   if (ibsta & REM ) fprintf (stderr, " REM");
   if (ibsta & CIC ) fprintf (stderr, " CIC");
   if (ibsta & ATN ) fprintf (stderr, " ATN");
   if (ibsta & TACS) fprintf (stderr, " TACS");
   if (ibsta & LACS) fprintf (stderr, " LACS");
   if (ibsta & DTAS) fprintf (stderr, " DTAS");
   if (ibsta & DCAS) fprintf (stderr, " DCAS");

   fprintf (stderr, " >\n");
   fprintf (stderr, "iberr= %d", iberr);
```

```
    if (iberr == EDVR) fprintf (stderr,
        " EDVR <DOS Error>\n");
    if (iberr == ECIC) fprintf (stderr,
        " ECIC <Not CIC>\n");
    if (iberr == ENOL) fprintf (stderr,
        " ENOL <No Listener>\n");
    if (iberr == EADR) fprintf (stderr,
        " EADR <Address error>\n");
    if (iberr == EARG) fprintf (stderr,
        " EARG <Invalid argument>\n");
    if (iberr == ESAC) fprintf (stderr,
        " ESAC <Not Sys Ctrlr>\n");
    if (iberr == EABO) fprintf (stderr,
        " EABO <Op. aborted>\n");
    if (iberr == ENEB) fprintf (stderr,
        " ENEB <No GPIB board>\n");
    if (iberr == EOIP) fprintf (stderr,
        " EOIP <Async I/O in prg>\n");
    if (iberr == ECAP) fprintf (stderr,
        " ECAP <No capability>\n");
    if (iberr == EFSO) fprintf (stderr,
        " EFSO <File sys. error>\n");
    if (iberr == EBUS) fprintf (stderr,
        " EBUS <Command error>\n");
    if (iberr == ESTB) fprintf (stderr,
        " ESTB <Status byte lost>\n");
    if (iberr == ESRQ) fprintf (stderr,
        " ESRQ <SRQ stuck on>\n");
}
```

Macro Program Execution Sample

This section shows a macro program execution sample. The macro programs are installed under the following directories in the analyzer:

- Macros specific to a user:
  C:\Program Files\Tektronix\wca200a\Python\wca200a\measmacro\nonregistered
- Macros included in a option:
  C:\Program Files\Tektronix\wca200a\Python\wca200a\measmacro\registered

In the example below, the following macro folders are placed in these directories:

MacroTest1, MacroTest2, and MacroTest3 under the *nonregistered* directory
MacroTest1, MacroTest4, and MacroTest5 under the *registered* directory

The MacroTest1 macro folder contains macro commands test1, test2, and test3.

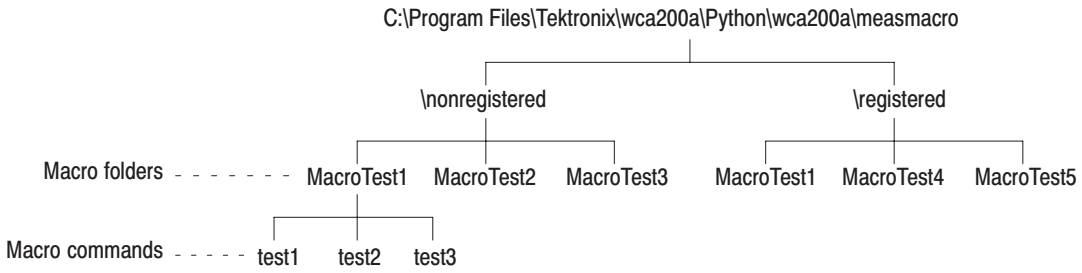


Figure 4–1: Saving the macro programs

Suppose that the following variables have been defined in the macro command test1:

LOW_LIMIT, HIGH_LIMIT (numeric parameters)
ERROR_MESSAGE (character string parameter)
RESULT (measurement results (numeric values))

The following is an example of sending and responding commands:

```
[Send]        PROG:CAT?      // Query the list of the macro program
[Response]    "NONREGISTERED.MACROTEST1",
              "NONREGISTERED.MACROTEST2",
              "NONREGISTERED.MACROTEST3",
              "REGISTERED.MACROTEST1",
              "REGISTERED.MACROTEST4",
              "REGISTERED.MACROTEST5"
[Send]        PROG:NAME "NONREGISTERED.MACROTEST1"
                        // Specify the macro program
[Send]        PROG:NUMB "LOW_LIMIT",1.5   // Set LOW_LIMIT to 1.5
[Send]        PROG:NUMB "HIGH_LIMIT",20   // Set HIGH_LIMIT to 20
[Send]        PROG:STR "ERROR_MESSAGE","Unsuccessful"
                        // Set ERROR_MESSAGE
[Send]        PROG:EXEC "TEST1"     // Run the macro command
[Send]        PROG:NUMB? "RESULT"  // Retrieve the results
[Response]    1.2345
[Send]        PROG:DEL       // Delete the macro program from memory
```

Appendices

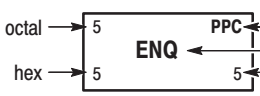
Appendix A: Character Charts

The ASCII and GPIB code chart is shown in Table A–1 on page A–2.

Table A–1: ASCII & GPIB code chart

B7 B6 B5 BITS B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 LA0 SP 20 32	60 LA16 0 30 48	100 TA0 @ 40 64	120 TA16 P 50 80	140 SA0 ` 60 96	160 SA16 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LL0 DC1 11 17	41 LA1 ! 21 33	61 LA17 1 31 49	101 TA1 A 41 65	121 TA17 Q 51 81	141 SA1 a 61 97	161 SA17 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 LA2 " 22 34	62 LA18 2 32 50	102 TA2 B 42 66	122 TA18 R 52 82	142 SA2 b 62 98	162 SA18 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 LA3 # 23 35	63 LA19 3 33 51	103 TA3 C 43 67	123 TA19 S 53 83	143 SA3 c 63 99	163 SA19 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 LA4 $ 24 36	64 LA20 4 34 52	104 TA4 D 44 68	124 TA20 T 54 84	144 SA4 d 64 100	164 SA20 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 LA5 % 25 37	65 LA21 5 35 53	105 TA5 E 45 69	125 TA21 U 55 85	145 SA5 e 65 101	165 SA21 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 LA6 & 26 38	66 LA22 6 36 54	106 TA6 F 46 70	126 TA22 V 56 86	146 SA6 f 66 102	166 SA22 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 LA7 ' 27 39	67 LA23 7 37 55	107 TA7 G 47 71	127 TA23 W 57 87	147 SA7 g 67 103	167 SA23 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 LA8 (28 40	70 LA24 8 38 56	110 TA8 H 48 72	130 TA24 X 58 88	150 SA8 h 68 104	170 SA24 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51 LA9) 29 41	71 LA25 9 39 57	111 TA9 I 49 73	131 TA25 Y 59 89	151 SA9 i 69 105	171 SA25 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 LA10 * 2A 42	72 LA26 : 3A 58	112 TA10 J 4A 74	132 TA26 Z 5A 90	152 SA10 j 6A 106	172 SA26 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 LA11 + 2B 43	73 LA27 ; 3B 59	113 TA11 K 4B 75	133 TA27 [5B 91	153 SA11 k 6B 107	173 SA27 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 LA12 , 2C 44	74 LA28 < 3C 60	114 TA12 L 4C 76	134 TA28 \ 5C 92	154 SA12 l 6C 108	174 SA28 ¦ 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 LA13 – 2D 45	75 LA29 = 3D 61	115 TA13 M 4D 77	135 TA29] 5D 93	155 SA13 m 6D 109	175 SA29 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 LA14 . 2E 46	76 LA30 > 3E 62	116 TA14 N 4E 78	136 TA30 ^ 5E 94	156 SA14 n 6E 110	176 ~ SA30 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 LA15 / 2F 47	77 UNL ? 3F 63	117 TA15 O 4F 79	137 UNT – 5F 95	157 SA15 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY



GPIB code (with ATN asserted)
ASCII character
decimal

Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Appendix B: GPIB Interface Specification

This appendix lists and describes the GPIB functions and messages the waveform generator implements.

Interface Functions

Table B–1 lists the GPIB interface functions this instrument implements. Each function is briefly described on page B–2.

Table B–1: GPIB interface function implementation

Interface function	Implemented subset	Capability
Source Handshake (SH)	SH1	Complete
Acceptor Handshake (AH)	AH1	Complete
Talker (T)	T6	Basic Talker, Serial Poll Unaddress if my-listen-address (MLA) No Talk Only mode
Listener (L)	L4	Basic Listener Unaddress if my talk address (MTA) No Listen Only mode
Service Request (SR)	SR1	Complete
Remote/Local (RL)	RL0	None
Parallel Poll (PP)	PP0	None
Device Clear (DC)	DC1	Complete
Device Trigger (DT)	DT0	None
Controller (C)	C0	None
Electrical Interface	E2	Three-state driver

- Source Handshake (SH). Enables a talking device to support the coordination of data transfer. The SH function controls the initiation and termination of data byte transfers.
- Acceptor Handshake (AH). Enables a listening device to coordinate data reception. The AH function delays data transfer initiation or termination until the listening device is ready to receive the next data byte.
- Talker (T). Enables a device to send device-dependent data over the interface. This capability is available only when the device is addressed to talk, and uses a one-byte address.
- Listener (L). Enables a device to receive device-dependent data over the interface. This capability is available only when the device is addressed to listen, and uses a one-byte address.
- Service Request (SR). Enables a device to assert an SRQ (Service Request) line to notify the controller when it requires service.
- Remote/Local (RL). Enables a device to respond to both the GTL (Go To Local) and LLO (Local Lock Out) interface messages.
- Parallel Poll (PP). Enables a device to respond to the following interface messages: PPC, PPD, PPE, and PPU, as well as to send out a status message when the ATN (Attention) and EOI (End or Identify) lines are asserted simultaneously.
- Device Clear (DC). Enables a device to be cleared or initialized, either individually, or as part of a group of devices.
- Device Trigger (DT). Enables a device to respond to the GET (Group Execute Trigger) interface message when acting as a listener.
- Controller (C). Enables a device that has this capability to send its address, universal commands, and addressed commands to other devices over the interface.
- Electrical Interface (E). Identifies the electrical interface driver type. The notation E1 means the electrical interface uses open collector drivers, E2 means the electrical interface uses three-state drivers.

Interface Messages

Table B–2 shows the standard interface messages that are supported by the analyzer.

Table B–2: Standard interface messages

Message	Type	Implemented
Device Clear (DCL)	UC	Yes
Local Lockout (LLO)	UC	No
Serial Poll Disable (SPD)	UC	Yes
Serial Poll Enable (SPE)	UC	Yes
Parallel Poll Unconfigure (PPU)	UC	No
Go To Local (GTL)	AC	Yes
Selected Device Clear (SDC)	AC	Yes
Group Execute Trigger (GET)	AC	No
Take Control (TCT)	AC	No
Parallel Poll Configure (PPC)	AC	No

UC: Universal command; AC: Address command

- Device Clear (DCL). Will clear (initialize) all devices on the bus that have a device clear function, whether or not the controller has addressed them.
- Local Lockout (LLO). Disables the return to local function.
- Serial Poll Disable (SPD). Changes all devices on the bus from the serial poll state to the normal operating state.
- Serial Poll Enable (SPE). Puts all bus devices that have a service request function into the serial poll enabled state. In this state, each device sends the controller its status byte, instead of its normal output, after the device receives its talk address on the data lines. This function may be used to determine which device sent a service request.
- Go To Local (GTL). Causes the listen-addressed device to switch from remote to local (front-panel) control.
- Select Device Clear (SDC). Clears or initializes all listen-addressed devices.
- Group Execute Trigger (GET). Triggers all applicable devices and causes them to initiate their programmed actions.
- Take Control (TCT). Allows the controller in charge to pass control of the bus to another controller on the bus.
- Parallel Poll Configure (PPC). Causes the listen-addressed device to respond to the secondary commands Parallel Poll Enable (PPE) and Parallel Poll Disable (PPD), which are placed on the bus following the PPC command. PPE enables a device with parallel poll capability to respond on a particular data line. PPD disables the device from responding to the parallel poll.

Appendix C: Factory Initialization Settings

The factory initialization settings provide you a known state for the analyzer. The *RST command returns the instrument settings to the factory defaults for the measurement mode specified with :INSTrument[:SELect]. Factory initialization sets values as shown in Table C–1 through C–11.

Table C–1: Factory initialization settings — IEEE common commands

Header	Default value
*ESE	0
*OPC	0
*SRE	0

Table C–2: Factory initialization settings — :CALCulate commands

Header	Default value
:CALCulate<x>:DLINe<y>	0
:CALCulate<x>:DLINe<y>:STATe	OFF
:CALCulate<x>:MARKer<y>:MODE	POSition
:CALCulate<x>:MARKer<y>:TRACe	MAIN
:CALCulate<x>:MARKer<y>:X	0
:CALCulate<x>:MARKer<y>:Y	0
:CALCulate<x>:VLINe<y>	0
:CALCulate<x>:VLINe<y>:STATe	OFF

Table C–3: Factory initialization settings — :CALibration commands

Header	Default value
:CALibration:AUTO	OFF

Table C–4: Factory initialization settings — :DISPlay commands

Header	Default value
:DISPlay:CCDF subgroup	
:DISPlay:CCDF:LINE:GAUSsian[:STATe]	ON
:DISPlay:CCDF:LINE:REFerence[:STATe]	OFF
:DISPlay:CCDF:X[:SCALe]:AUTO	ON
:DISPlay:CCDF:X[:SCALe]:MAXimum	15 dB
:DISPlay:CCDF:X[:SCALe]:OFFSet	0 dB
:DISPlay:CCDF:Y[:SCALe]:MAXimum	1E–7
:DISPlay:CCDF:Y[:SCALe]:MINimum	100%
:DISPlay:OVIew subgroup	
:DISPlay:OVIew:FORMat	WAVeform
:DISPlay:OVIew:SGRam:COLor[:SCALe]:OFFSet	–100 dBm
:DISPlay:OVIew:SGRam:COLor[:SCALe]:RANGe	100 dB
:DISPlay:OVIew:SGRam:X[:SCALe]:OFFSet	1.495 GHz
:DISPlay:OVIew:SGRam:X[:SCALe]:SPAN	10 MHz
:DISPlay:OVIew:SGRam:Y[:SCALe]:OFFSet	0
:DISPlay:OVIew:SGRam:Y[:SCALe]:PLINe	1
:DISPlay:OVIew:WAVeform:X[:SCALe]:OFFSet	–320 µs
:DISPlay:OVIew:WAVeform:X[:SCALe]:PDIVision	32 µs/div
:DISPlay:OVIew:WAVeform:Y[:SCALe]:OFFSet	–100 dBm
:DISPlay:OVIew:WAVeform:Y[:SCALe]:PDIVision	100 dB
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:OFFSet	–100 dBm
:DISPlay:OVIew:ZOOM:COLor[:SCALe]:RANGe	100 dB
:DISPlay:OVIew:ZOOM:X[:SCALe]:OFFSet	1.495 GHz
:DISPlay:OVIew:ZOOM:X[:SCALe]:SPAN	10 MHz
:DISPlay:OVIew:ZOOM:Y[:SCALe]:OFFSet	0
:DISPlay:OVIew:ZOOM:Y[:SCALe]:PLINe	1
:DISPlay:PULSe subgroup	
:DISPlay:PULSe:MVIew:RESult:CHPower	OFF
:DISPlay:PULSe:MVIew:RESult:DCYCle	OFF
:DISPlay:PULSe:MVIew:RESult:EBWidth	OFF
:DISPlay:PULSe:MVIew:RESult:FREQuency	OFF
:DISPlay:PULSe:MVIew:RESult:OBWidth	OFF
:DISPlay:PULSe:MVIew:RESult:OORatio	OFF

Table C–4: Factory initialization settings — :DISPlay commands (Cont.)

Header	Default value
:DISPlay:PULSe:MVIew:RESult:PERiod	OFF
:DISPlay:PULSe:MVIew:RESult:PHASe	OFF
:DISPlay:PULSe:MVIew:RESult:PPOWer	OFF
:DISPlay:PULSe:MVIew:RESult:RIPPle	OFF
:DISPlay:PULSe:MVIew:RESult:WIDTh	ON
:DISPlay:PULSe:SVIew:FORMat	WIDTh
:DISPlay:PULSe:SVIew:GUIDelines	ON
:DISPlay:PULSe:SVIew:RANGe	ADAPtive
:DISPlay:PULSe:SVIew:RESult	SINGle
:DISPlay:PULSe:SVIew:SELect	0
:DISPlay:SPECtrum subgroup	
:DISPlay:SPECtrum:BMARker:STATe	ON
:DISPlay:SPECtrum:GRATicule:GRID	FIX
:DISPlay:SPECtrum:MLINe:AMPLitude:INTerval	0 dB
:DISPlay:SPECtrum:MLINe:AMPLitude:OFFSet	0 dBm
:DISPlay:SPECtrum:MLINe:AMPLitude[:STATe]	OFF
:DISPlay:SPECtrum:MLINe:ANNotation[:STATe]	ON
:DISPlay:SPECtrum:MLINe:FREQuency:INTerval	0 Hz
:DISPlay:SPECtrum:MLINe:FREQuency:OFFSet	Center frequency
:DISPlay:SPECtrum:MLINe:FREQuency[:STATe]	OFF
:DISPlay:SPECtrum:X[:SCALe]:OFFSet	1.495 GHz
:DISPlay:SPECtrum:X[:SCALe]:PDIVision	1 MHz/div
:DISPlay:SPECtrum:Y[:SCALe]:OFFSet	–100 dBm
:DISPlay:SPECtrum:Y[:SCALe]:PDIVision	10 dB/div
:DISPlay:TFRequency subgroup	
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:OFFSet	–100 dBm
:DISPlay:TFRequency:SGRam:COLor[:SCALe]:RANGe	100 dB
:DISPlay:TFRequency:SGRam:MLINe:ANNotation[:STATe]	ON
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:INTerval	0 Hz
:DISPlay:TFRequency:SGRam:MLINe:FREQuency:OFFSet	Center frequency
:DISPlay:TFRequency:SGRam:MLINe:FREQuency[:STATe]	OFF
:DISPlay:TFRequency:SGRam:MLINe:TIME:INTerval	0 s

Table C–4: Factory initialization settings — :DISPlay commands (Cont.)

Header	Default value
:DISPlay:TFRequency:SGRam:MLINe:TIME:OFFSet	10 ms
:DISPlay:TFRequency:SGRam:MLINe:TIME[:STATe]	OFF
:DISPlay:TFRequency:SGRam:X[:SCALe]:OFFSet	1.495 GHz
:DISPlay:TFRequency:SGRam:X[:SCALe]:SPAN	10 MHz
:DISPlay:TFRequency:SGRam:Y[:SCALe]:OFFSet	0
:DISPlay:TFRequency:SGRam:Y[:SCALe]:PLINe	1
:DISPlay[:VIEW] subgroup	
:DISPlay[:VIEW]:BRIGhtness	100
:DISPlay[:VIEW]:FORMat	V1S (SANORMAL) MULTitude (Other than above)
:DISPlay:WAVeform subgroup	
:DISPlay:WAVeform:X[:SCALe]:OFFSet	–320 µs
:DISPlay:WAVeform:X[:SCALe]:PDIVision	32 µs/div
:DISPlay:WAVeform:Y[:SCALe]:OFFSet	0
:DISPlay:WAVeform:Y[:SCALe]:PDIVision	0

Table C–5: Factory initialization settings — :FORMat commands

Header	Default value
:FORMat:BORDer	NORMal
:FORMat[:DATA]	REAL,32

Table C–6: Factory initialization settings — :INITiate commands

Header	Default value
:INITiate:CONTinuous	OFF

Table C–7: Factory initialization settings — :INPut commands

Header	Default value
:INPut:ATTenuation	20 dB
:INPut:ATTenuation:AUTO	ON
:INPut:MAXLevel	0 dB
:INPut:MIXer	–25 dBm

Table C–8: Factory initialization settings — :SENSe commands

Header	Default value
[:SENSe]:ACPower subgroup	
[:SENSe]:ACPower:BANDwidth|BWIDth:ACHannel	1 MHz
[:SENSe]:ACPower:BANDwidth|BWIDth:INTegration	1 MHz
[:SENSe]:ACPower:CSPacing	1.4 MHz
[:SENSe]:ACPower:FILTer:TYPE	NYQuist
[:SENSe]:ACPower:FILTer:COEFficient	0.5
[:SENSe]:ADEMod subgroup	
[:SENSe]:ADEMod:BLOCk	0
[:SENSe]:ADEMod:CARRier:OFFSet	0
[:SENSe]:ADEMod:CARRier:SEARch	ON
[:SENSe]:ADEMod:FM:THReshold	–100 dB
[:SENSe]:ADEMod:LENGth	2048
[:SENSe]:ADEMod:MODulation	OFF
[:SENSe]:ADEMod:OFFSet	0
[:SENSe]:ADEMod:PM:THReshold	–100 dB
[:SENSe]:AVERage subgroup	
[:SENSe]:AVERage:COUNt	20
[:SENSe]:AVERage[:STATe]	OFF
[:SENSe]:AVERage:TCONtrol	EXPonential
[:SENSe]:BSIZe subgroup	
[:SENSe]:BSIZe	2
[:SENSe]:CCDF subgroup	
[:SENSe]:CCDF:BLOck	0
[:SENSe]:CCDF:THReshold	–150 dBm

Table C–8: Factory initialization settings — :SENSe commands (Cont.)

Header	Default value
[:SENSe]:CFRequency subgroup	
[:SENSe]:CFRequency:CRESolution	1 Hz
[:SENSe]:CHPower subgroup	
[:SENSe]:CHPower:BANDwidth|BWIDth:INTegration	2 MHz
[:SENSe]:CHPower:FILTer:COEFficient	0.5
[:SENSe]:CHPower:FILTer:TYPE	NYQuist
[:SENSe]:CNRatio subgroup	
[:SENSe]:CNRatio:BANDwidth|BWIDth:INTegration	1 MHz
[:SENSe]:CNRatio:BANDwidth|BWIDth:NOISe	1 MHz
[:SENSe]:CNRatio:FILTer:COEFficient	0.5
[:SENSe]:CNRatio:FILTer:TYPE	NYQuist
[:SENSe]:CNRatio:OFFSet	3 MHz
[:SENSe]:CORRection subgroup	
[:SENSe]:CORRection:OFFSet[:MAGNitude]	0
[:SENSe]:CORRection:OFFSet:FREQuency	0
[:SENSe]:CORRection[:STATe]	OFF
[:SENSe]:CORRection:X:SPACing	LINear
[:SENSe]:CORRection:Y:SPACing	LOGarithmic
[:SENSe]:EBWidth subgroup	
[:SENSe]:EBWidth:XDB	–30 dB
[:SENSe]:FEED subgroup	
[:SENSe]:FEED	RF
[:SENSe]:FREQuency subgroup	
[:SENSe]:FREQuency:CENTer	1.5 GHz
[:SENSe]:FREQuency:CENTer:STEP:AUTO	ON
[:SENSe]:FREQuency:CENTer:STEP[:INCRement]	100 kHz
[:SENSe]:FREQuency:CTABle[:SELect]	None
[:SENSe]:FREQuency:SPAN	10 MHz
[:SENSe]:FREQuency:STARt	1.495 GHz
[:SENSe]:FREQuency:STOP	1.505 GHz
[:SENSe]:OBWidth subgroup	
[:SENSe]:OBWidth:PERCent	99%

Table C–8: Factory initialization settings — :SENSe commands (Cont.)

Header	Default value
[:SENSe]:PULSe subgroup	
[:SENSe]:PULSe:BLOCk	0
[:SENSe]:PULSe:CHPower:BANDwidth|:BWIDth:INTegration	1 MHz
[:SENSe]:PULSe:CRESolution	1 kHz
[:SENSe]:PULSe:EBWidth:XDB	–30 dB
[:SENSe]:PULSe:FILTer:COEFficient	0.35
[:SENSe]:PULSe:FILTer:BANDwidth|BWIDth	3.6 MHz
[:SENSe]:PULSe:FILTer:MEASuerment	OFF
[:SENSe]:PULSe:FILTer:OBWidth:PERcent	90%
[:SENSe]:PULSe:PTOFfset	0
[:SENSe]:PULSe:THReshold	–3 dBc
[:SENSe]:ROSCillator subgroup	
[:SENSe]:ROSCillator:SOURce	INTernal
[:SENSe]:SPECtrum subgroup	
[:SENSe]:SPEctrum:AVERage:COUNt	20
[:SENSe]:SPEctrum:AVERage[:STATe]	OFF
[:SENSe]:SPEctrum:AVERage:TYPE	RMS
[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution]	50 kHz
[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution]:AUTO	ON
[:SENSe]:SPECtrum:BANDwidth|BWIDth:STATe	ON
[:SENSe]:SPECtrum:DETector[:FUNCtion]	POSitive
[:SENSe]:SPECtrum:FILTer:COEFficient	0.5
[:SENSe]:SPECtrum:FILTer:TYPE	NYQuist
[:SENSe]:SPECtrum:FFT:ERESolution	OFF
[:SENSe]:SPECtrum:FFT:LENGth	1024
[:SENSe]:SPECtrum:FFT:STARt	1024
[:SENSe]:SPECtrum:FFT:WINDow[:TYPE]	BH4B
[:SENSe]:SPECtrum:FRAMe	0
[:SENSe]:SPECtrum:MEASurement	OFF
[:SENSe]:SPECtrum:ZOOM:BLOCk	0
[:SENSe]:SPECtrum:ZOOM:FREQuency:CENTer	Center frequency
[:SENSe]:SPECtrum:ZOOM:FREQuency:WIDTh	Span

Table C–8: Factory initialization settings — :SENSe commands (Cont.)

Header	Default value
[:SENSe]:SPECtrum:ZOOM:LENGth	7680
[:SENSe]:SPECtrum:ZOOM:OFFSet	0
[:SENSe]:SPURious subgroup	
[:SENSe]:SPURious[:THReshold]:EXCursion	3 dB
[:SENSe]:SPURious[:THReshold]:IGNore	0 Hz
[:SENSe]:SPURious[:THReshold]:SIGNal	–20 dBm
[:SENSe]:SPURious[:THReshold]:SPURious	–70 dB
[:SENSe]:TRANsient subgroup	
[:SENSe]:TRANsient:BLOCk	0
[:SENSe]:TRANsient:ITEM	OFF
[:SENSe]:TRANsient:LENGth	2048
[:SENSe]:TRANsient:OFFSet	0

Table C–9: Factory initialization settings — :STATus commands

Header	Default value
:STATus:OPERation:ENABle	0
:STATus:QUEStionable:ENABle	0
:SYSTem:QUESTionable[:EVENt]	0

Table C–10: Factory initialization settings — :TRACe commands

Header	Default value
:TRACe<x>:MODE	NORMal
:TRACe<x>:DDETector	MAXimum
:TRACe<x>:AVERage:COUNt	20

Table C–11: Factory initialization settings — :TRIGger commands

Header	Default value
:TRIGger[:SEQuence]:LEVel:IF	50%
:TRIGger[:SEQuence]:MODE	AUTO
:TRIGger[:SEQuence]:POSition	50%
:TRIGger[:SEQuence]:SLOPe	Rise
:TRIGger[:SEQuence]:SOURce	IF

Table C–12: Factory initialization settings — :UNIT commands

Header	Default value
:UNIT:ANGLe	DEG

Appendix D: Setting Range

This section lists the setting range of the horizontal and vertical scale for the views, and of RBW (Resolution Bandwidth).

Display Format and Scale

Table D–1: Display format and scale

Display format	Horizontal range	Vertical range
Spectrum	0 Hz to 3 GHz (RSA2203A) 0 Hz to 8 GHz (RSA2208A)	–200 to +100 dBm
Spectrogram	0 Hz to 3 GHz (RSA2203A) 0 Hz to 8 GHz (RSA2208A)	Frame –499 to 0
Time domain view	$-(T_f \times N_f)$ to 0 s *	–200 to +100 dBm (Amplitude) –30 to +30 V (I/Q level) –300 to +300% (AM) –38.4 to +38.4 MHz (FM/FVT) –675 to +675 deg. (PM)
CCDF	0 to 15.01 dB	10^{-9} to 100%

* **T_f: Frame time; N_f: Frame number**

RBW

The RBW setting range depends on span as shown in Table D–2.

Table D–2: RBW setting range

Span (Hz)	Default value (Hz) /[Number of samples]		Minimum value (Hz) /[Number of samples]		Maximum value (Hz) /[Number of samples]	
50 to 100	2	[1024]	1	[2048]	10	[128]
120 to 200	5	[512]	1	[4096]	20	[128]
250 to 500	10	[1024]	1	[8192]	50	[128]
600 to 1 k	20	[1024]	1	[16384]	100	[128]
1.2 k to 2 k	50	[512]	2	[16384]	200	[128]
2.5 k to 5 k	100	[1024]	5	[16384]	500	[128]
6 k to 10 k	100	[2048]	10	[16384]	1 k	[128]
12 k to 20 k	200	[2048]	20	[16384]	2 k	[128]
30 k to 50 k	300	[4096]	50	[16384]	5 k	[128]
60 k to 100 k	500	[4096]	100	[16384]	10 k	[128]
120 k to 200 k	1 k	[4096]	200	[16384]	20 k	[128]
250 k to 500 k	2 k	[2048]	500	[16384]	50 k	[128]
600 k to 1 M	5 k	[2048]	1 k	[16384]	100 k	[128]
1.2 M to 2 M	10 k	[4096]	1 k	[32768]	200 k	[128]
2.5 M to 5 M	20 k	[4096]	1 k	[65536]	500 k	[256]
6 M to 10 M	50 k	[2048]	1 k	[65536]	1 M	[128]
15 M	80 k	[4096]	2 k	[65536]	2 M	[256]
20 M to 40 M	100 k	[1024*N]	10 k	[8192*N]	2 M	[64*N]
50 M to 80 M	300 k	[512*N]	10 k	[8192*N]	2 M	[64*N]
100 M to 150 M	500 k	[256*N]	10 k	[8192*N]	10 M	[64*N]
200 M to 400 M	1 M	[128*N]	10 k	[8192*N]	10 M	[64*N]
500 M to 800 M	2 M	[128*N]	20 k	[4096*N]	10 M	[64*N]
1 G to 1.5 G	5 M	[128*N]	50 k	[2048*N]	20 M	[64*N]
2 G to 3 G	10 M	[128*N]	100 k	[1024*N]	30 M	[64*N]

*** N: Number of multi-frames, that is the value rounded off [(span)/(10 MHz)] to the positive infinity.**

Appendix E: SCPI Conformance Information

All commands in the RSA2200 Series analyzers are based on SCPI Version 1999.0. Table E–1 lists the commands that are defined in the SCPI 1999.0 Standard. The other commands not listed in the table are not defined in the SCPI 1999.0 Standard.

Table E–1: SCPI 1999.0-defined commands

Command group	Command		
IEEE common	*CAL?		
	*CLS		
	*ESE		
	*ESR?		
	*IDN?		
	*OPC		
	*RST		
	*SRE		
	*STB?		
	*TST?		
	*WAI		
:ABORt	:ABORt		
:CALibration	:CALibration	[:ALL]?	
		:AUTO	
:HCOPy	:HCOPy	:DESTination	
		[:IMMediate]	
:INITiate	:INITiate	:CONTinuous	
		[:IMMediate]	
		:RESTart	
:INPut	:INPut	:ATTenuation	
			:AUTO
		:COUPling	
:INSTrument	:INSTrument	:CATalog	
		[:SELect]	
:MMEMory	:MMEMory	:COPY	
		:DELete	
		:NAME	

Table E–1: SCPI 1999.0-defined commands (Cont.)

Command group	Command				
:PROGram	:PROGram	:CATalog?			
		[:SELected]	:DELete	[:SELected]	
			:EXECute		
			:NAME		
		:NUMBer			
		:STRing			
:SENSe	[:SENSe]	:FREQuency	:CENTer		
				:STEP	:AUTO
					[:INCrement]
			:SPAN		
			:STARt		
			:STOP		
		:ROSCillator	:SOURce		
:STATus	:STATus	:OPERation	:CONDition?		
			:ENABle		
			[:EVENt]?		
			NTRansition		
			PTRansition		
		:PRESet			
		:QUEStionable	:CONDition?		
			:ENABle		
			[:EVENt]?		
			NTRansition		
			PTRansition		
:SYSTem	:SYSTem	:DATE			
		:ERRor	:ALL?		
			:CODE	:ALL?	
				[:NEXT]?	
			:COUNt?		
			[:NEXT]?		
		:KLOCk			
		:PRESet			
		:TIME			
		:VERSion?			

Table E–1: SCPI 1999.0-defined commands (Cont.)

Command group	Command		
:TRIGger	:TRIGger	[:SEQuence]	:MODE
			:POSition
			:SLOPe
			:SOURce
:UNIT	:UNIT	:ANGLe	

Glossary and Index

Glossary

AM (Amplitude Modulation)
The process, or result of a process, in which the amplitude of a sine wave (the carrier) is varied in accordance with the instantaneous voltage of a second electrical signal (the modulating signal).

ASCII
Acronym for the American Standard Code for Information Interchange. Controllers transmit commands to the analyzer using ASCII character encoding.

Backus-Naur Form (BNF)
A standard notation system for command syntax diagrams. The syntax diagrams in this manual use BNF notation.

Controller
A computer or other device that sends commands to and accepts responses from the analyzer.

EVM (Error Vector Magnitude)
The magnitude of an error of an actual signal relative to an ideal signal in a constellation display.

FM (Frequency Modulation)
The process, or result of a process, in which the frequency of an electrical signal (the carrier) is varied in accordance with some characteristic of a second electrical signal (the modulating signal or modulation).

GPIB
Acronym for General Purpose Interface Bus, the common name for the communications interface system defined in IEEE Std 488.

IEEE
Acronym for the Institute for Electrical and Electronic Engineers.

PM (Pulse Modulation)
The process, or result of a process, in which the amplitude, phase, or duration of a pulse train (the carrier) is varied in accordance with some characteristic of a second electrical signal (the modulating signal or modulation).

Index

H

I

M

P

Q

R

S

T

U